OPTIMIZATION OF HETEROGENEOUS UAV COMMUNICATIONS USING THE MULTIOBJECTIVE QUADRATIC ASSIGNMENT PROBLEM

THESIS

Mark P. Kleeman, First Lieutenant, USAF

AFIT/GCE/ENG/04-04

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCE/ENG/04-04

OPTIMIZATION OF HETEROGENEOUS UAV COMMUNICATIONS USING THE MULTIOBJECTIVE QUADRATIC ASSIGNMENT PROBLEM

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Mark P. Kleeman, A.A., A.S., B.S.

First Lieutenant, USAF

March, 2004

AFIT/GCE/ENG/04-04

OPTIMIZATION OF HETEROGENEOUS UAV COMMUNICATIONS USING THE MULTIOBJECTIVE QUADRATIC ASSIGNMENT PROBLEM

Mark P. Kleeman, A.A., A.S., B.S.
First Lieutenant, USAF

Approved:

/signed/	10 Mar 2004
Dr. Gary B. Lamont Chairman	Date
/signed/	10 Mar 2004
Dr. Gilbert L. Peterson Member	Date
/signed/	10 Mar 2004
Dr. Meir Pachter Member	Date

Acknowledgements

When a person achieves a major goal in their life, it is rarely achieved without the help and support of others. This accomplishment for me is no exception. First, I'd like to thank several educators who have been an inspiration to me along the way. The one instructor who stands out in my mind that truly inspired me to reach heights that I didn't know I could achieve is Dr. Bob Myers from Northern Michigan University. I doubt I would be where I am today if it weren't for the valuable lessons he taught me. I would also like to thank Dr. Gyula Mago from the University of North Carolina for teaching me there are more important things to life than an education. And of course, this thesis would not have become a reality had it not been for the guidance and wisdom of my thesis advisor, Dr. Gary Lamont. His guidance and knowledge were a vital part in my educational experience at AFIT. Without him, I doubt I would have learned half as much as I did.

I'd like to also thank the friends that I've made here at AFIT. Going through this with them helped to make the long hours more bearable. I'd like to thank Joshua Corner for the great conversation and friendship that we forged, Zach Gray for all the sports talk and debates, Rick Day for his PhD "enlightenment", and the other students who touched my life one way or another.

But most importantly I'd like to thank my family for the incredible support and love they've shown me. First, I'd like to thank my parents for giving my a wonderful childhood full of joy and teaching me how to work hard to reach my goals. I'd also like to thank my kids, who could always erase the memory of a bad day at school with a smile and a hug. But the biggest amount of thanks goes to my wife who has had to run the family when school has bogged me down. Her love and generosity are greatly appreciated and I doubt I'll ever be able to repay her for all that she has done for me over this last year and a half. I love you all dearly. But most importantly, I'd like to give thanks and praise to God, who has given me all the wonderful things in my life and has blessed me with my family and friends. With Him, all things are possible.

Mark P. Kleeman

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation		Page
mGA	Messy Genetic Algorithm	4-1
fmGA	Fast Messy Genetic Algorithm	4-1
PCI	Probabilistically Complete Initialization	4-1
MOMGA	Multiobjective Messy Genetic Algorithm	4-6
PEI	Partially Enumerative Initialization	C-5

Abstract

The Air Force has placed a high priority on developing new and innovative ways to use Unmanned Aerial Vehicles (UAVs). The Defense Advanced Research Projects Agency (DARPA) currently funds many projects that deal with the advancement of UAV research. The ultimate goal of the Air Force is to use UAVs in operations that are highly dangerous to pilots, mainly the suppression of enemy air defenses (SEAD). With this goal in mind, formation structuring of autonomous or semi-autonomous UAVs is of future importance.

This particular research investigates the optimization of heterogeneous UAV multi-channel communications in formation. The problem maps to the multiobjective Quadratic Assignment Problem (mQAP). Optimization of this problem is done through the use of a Multiobjective Evolutionary Algorithm (MOEA) called the Multiobjective Messy Genetic Algorithm - II (MOMGA-II). Experimentation validates the attainment of an acceptable Pareto Front for a variety of mQAP benchmarks. It was observed that building block size can affect the location vectors along the current Pareto Front. The competitive templates used during testing perform best when they are randomized before each building block size evaluation. This tuning of the MOMGA-II parameters creates a more effective algorithm for the variety of mQAP benchmarks, when compared to the initial experiments. Thus this algorithmic approach would be useful for Air Force decision makers in determining the placement of UAVs in formations.

OPTIMIZATION OF HETEROGENEOUS UAV COMMUNICATIONS USING THE MULTIOBJECTIVE QUADRATIC ASSIGNMENT PROBLEM

1. Introduction

Unmanned Aerial Vehicles (UAVs) are quickly moving into the forefront of military aviation. In fact, many of the current leaders, including the current Air Force Chief of Staff, General Jumper, are expressing their desire for an increased role for UAVs [9]. Even congress has weighed in on the matter, stating that one-third of the deep strike force aircraft should be unmanned by 2010 [30]. Currently, UAV development is focusing on more traditional aircraft and delivery mechanisms, but some research also includes miniature UAVs and autonomous, cooperative control [83, 101]. This particular investigation emphasizes optimization of abstract UAV communications.

1.1 Problem Statement

Currently, the primary roles for UAVs are reconnaissance and surveillance missions over the battle field. But, by the year 2010, the Air Force hopes to perform suppression of enemy air defense (SEAD) missions with the unmanned combat aerial vehicle (UCAV) [30]. Further in the future, the Air Force wants UAVs, flying in large groups, to play a bigger role over the field of battle. One viable scenario is having a heterogenous group of UAVs flying jointly to meet an objective. There could be some in the group whose job is reconnaissance and reporting the information to the UCAV. The UCAV's goal is taking out a target when it is located by one of the other UAVs. In addition, "fighter" UAVs may be present, who defend the group of UAVs from enemy aircraft. Figures 1.1 and 1.2 show examples heterogenous UAVs flying in formation.

While location in the formation for their particular part of the mission is important, they also need to be in a position where they can communicate effectively with the UAVs that they need. For example, the reconnaissance UAVs need to communicate coordinates to the combat UCAVs, to enable them to find their target. The fighter UAVs need to

Figure 1.1 An example of a heterogeneous UAV formation

communicate with all of the other UAVs when they sense approaching enemy aircraft, so that the group can take evasive action. And the UCAVs need to communicate when they have no more munitions left. All of these flows of communication can also dictate where the best location in the group may be for each UAV.

Specifically stated, the problem analyzed in this research consists of a heterogeneous mix of 10, 20, and 30 UAVs, all flying in a fixed formation with fixed communication rates to the other UAVs. This problem is easily extended to include more UAVs, but 10 - 30 was chosen based on the test suite used.

1.2 Sponsors

This research is sponsored by the Information Directorate (IF) and the Air Vehicles (VA) Directorate, Air Force Research Laboratory (AFRL), Wright Patterson Air Force Base, Ohio. The mission statement of AFRL/IF is: "The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements

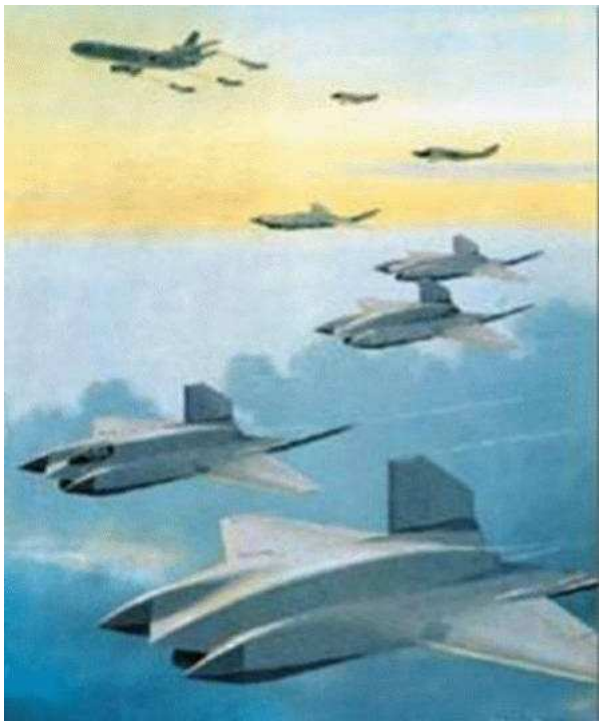

Figure 1.2 Another example of a heterogeneous UAV formation

for Information Dominance and its transition to air and space systems to meet warfighter needs" [2]. The research presented here supports this mission through the development of an effective algorithm that limits the total propagation time of communications in a heterogenous mix of UAVs flying in a formation. Specific points of contact include Dr. Robert L. Ewing (AFRL/IFTA) and Mr. Bruce T. Clough (AFRL/VACC).

1.3 Goals, Objectives, and Approach

The ultimate goal of this research is to find an algorithm that effectively and efficiently limits the aggregate amount of communication propagation time among a fixed formation of 10 - 30 heterogeneous UAVs. This goal is decomposed into two objectives:

1. Develop an effective algorithm to solve the problem
2. Attempt to improve the algorithm's efficiency while maintaining the effectiveness

1.3.1 Development of an Effective Search Algorithm. The first objective is to find an algorithm that effectively solves the communication problem. But what is considered effective, especially if there are no optimal solutions known for the problem? For this research, an effective algorithm is one that finds better results than most of the other published results. Several sub-objectives are required to validate this effectiveness:

1. Mapping of the UAV problem to a mathematical formulation and analysis of structure.
2. If feasible, use a deterministic algorithm to solve low dimensional problem optimally.
3. Find a good stochastic algorithm to solve problems of high UAV dimension.
4. Use optimal results, if available, to compare results of stochastic method.
5. Compare stochastic results with other results from literature

1.3.2 Efficiency Objectives. The second objective is to improve the efficiency of the algorithm. In order to realize this objective, several steps are taken. These sub-objectives are:

1. Use initial runs as a baseline for future runs and to test improvement.
2. Look for bottlenecks in the code and try to alleviate them when applicable.
3. Apply parallel processing in an attempt to speed up the algorithm.

1.4 Thesis Overview

Chapter 2 gives background information on the problem and on the mQAP. Chapter 3 supplies the background information for the MOEA. Chapter 4 describes the MOMGA-II algorithm and its development. Chapter 5 goes into detail about the software design. Chapter 6 describes the design of experiments. Chapter 7 discusses the results and presents an analysis of the data. Chapter 8 presents the conclusions drawn from this research as well as future work.

2. Multiobjective Quadratic Assignment Problem

2.1 Introduction

Computers are powerful computational tools and researchers use them to solve many complex problems. These problems are typically grouped into more generalized problem classes. These include the the travelling salesmen problem (TSP), the graph coloring problem, and the vehicle routing problem (VRP). Problems are also grouped into meta classifications based on their complexity. They include polynomial (P), nondeterministic polynomial (NP), and (NP-Complete) problems. Section 2.2 states the problem we are trying to solve. Section 2.3 gives a description of the single objective QAP, its origins, and its applications. Section 2.4 discusses the mQAP and how it is related to this research.

2.2 Problem Description

Currently, the Air Force uses UAVs for reconnaissance missions over the field of battle. But they are looking to expand the UAV role in the near future. By the year 2010, the Air Force hopes to perform SEAD missions with the UCAV [30]. Further in the future, the Air Force wants UAVs, flying in large groups, to play a bigger role in the air. One viable scenario is having a heterogenous group of UAVs flying jointly to meet an objective. There could be some in the group whose job is reconnaissance and reporting the information to the UCAV. The UCAV's goal is taking out a target when it is located by one of the other UAVs. In addition, "fighter" UAVs may be present, who defend the group of UAVs from enemy aircraft.

In a large, heterogenous group, such as this one, a UAVs position, with respect to the other UAVs is important. For example, it would be best to place fighter UAVs around the outside of the group in order to protect the group as a whole from enemy aircraft. It would also be advantageous to have the reconnaissance planes nearer to the ground in order to allow them to have a full field of view that isn't obstructed by other aircraft.

While location in the formation for their particular part of the mission is important, they also need to be in a position where they can communicate effectively with the UAVs that they need to. For example, the reconnaissance UAVs will need to communicate

coordinates to the combat UCAVs, to enable them to find their target. The fighter UAVs will need to communicate with all of the other UAVs when they sense approaching enemy aircraft, so that the group can take evasive action. And the UCAVs need to communicate when they have no more munitions left. All of these flows of communication can also dictate where the best location in the group may be for each UAV.

The UAV communication and mission success problem is a natural extension of the multiobjective Quadratic Assignment Problem (mQAP). The mQAP comes from the quadratic assignment problem (QAP) and was introduced by Knowles and Corne [71]. The following sections go into more detail about both the QAP and mQAP.

2.3 Quadratic Assignment Problem (QAP)

The QAP is a Combinatorial Optimization Problem (COP). The definition of a general COP is as follows:

Definition 1 (General COP): *Let B be a finite set called the* ground set*. The objective of the combinatorial optimization problem is to find the minimum cost element in the set of feasible solutions $X \subseteq 2^B$, i.e.,*

$$min\{c(x) : x \in X\}$$

where $c : X \rightarrow \Re$ denotes a cost mapping [77]. □

The QAP is a minimization problem. Section 2.3.1 gives a brief history of the QAP. It also gives the definition of the QAP. Since its inception, many problems have been mapped to the QAP. Section 2.3.2 mentions a few of these problems. Finally, Section 2.3.3 lists a few of the methods that have been used to solve the QAP.

2.3.1 History and Definition. Koopmans and Beckmann formulated the QAP in 1957 [76]. They used it to model the cost of interplant transportation among several industrial locations. They found that the linear assignment problem, which ignored the cost of transportation between locations, was too constrained.

2.3.1.1 Literal QAP Definition. A QAP consists of a finite number of locations that have fixed distances between each location. In addition, there are an equal number of facilities that need to be mapped to each location. Each facility has a fixed flow to every other facility. To arrive at a solution, every facility is mapped to a location and the flow to every facility is multiplied by the distance to every location. The cost is the summation of all of these products. The best answer is the lowest cost generated.

Most researchers represent the QAP input as two matrices. Presented below is an example of a five location, five facility QAP instance. Table 2.1 displays the matrix of distances between locations. Table 2.2 shows a matrix of the facilities and their respective flows. Notice how every location needs a distance between it and the other distances, but the facilities can have no flow to another facility. Also note that by setting all the flows to one, the QAP is simplified to the linear assignment problem

Table 2.1 **Example of a QAP matrix relating 5 locations and their distances apart**

Location	1	2	3	4	5
1	0	10	15	10	5
2	10	0	10	12	15
3	15	10	0	10	20
4	10	12	10	0	7
5	5	15	20	7	0

Table 2.2 **Example of a QAP matrix relating 5 facilities and their flows to each other**

Facility	1	2	3	4	5
1	0	2	5	0	2
2	2	0	0	4	5
3	5	0	0	4	0
4	0	4	4	0	0
5	2	5	0	0	0

A graphical representation of the QAP example is shown in figure 2.1.

2.3.1.2 Mathematical QAP Definition 2.1. One popular mathematical definition of the QAP is as follows:

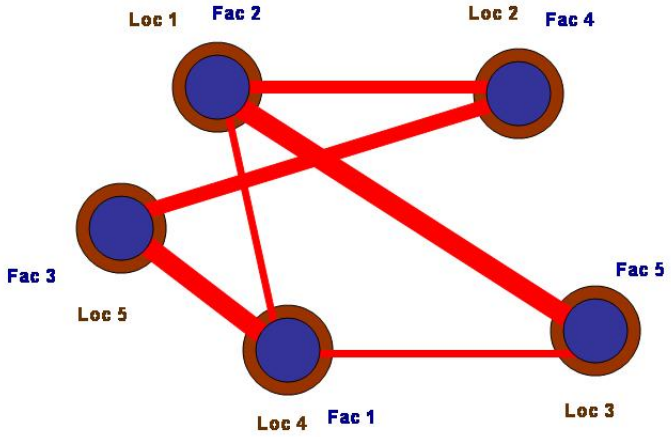

Figure 2.1 Graphical example of a QAP representation for 5 locations and 5 facilities

$$minC(\pi) =_{\pi \in P(n)}^{min} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi_i \pi_j} \tag{2.1}$$

where n is the number of objects/locations, a_{ij} is the distance between location i and location j, b_{ij} is the flow from object i to object j, and π_i gives the location of object i in permutation $\pi \in P(n)$ where $P(n)$ is the QAP search space, which is the set of all permutations of $\{1, 2, \ldots, n\}$ [72].

This problem is not only NP-hard and NP-hard to approximate, but is almost intractable [16]. It is generally considered to be impossible to solve optimally any QAP that has 20 instances or more within a reasonable time frame [16, 95].

2.3.2 Applications. Many applications have been mapped to the QAP. The facility location problem [76] has already been discussed. Other applications that have been mapped to the QAP include:

- Interconnection networks [82]
- Backboard wiring problems [93]
- Network software switches supporting internet telephony [110]

- Digital signal processor (DSP) memory layout [124]
- Floor planning for Very Large System Integration (VLSI) design [125]

In addition, more applications are listed in [16]. These include applications in sports, chemistry, and archeology to name a few.

Some in depth analysis of the QAP has been done by some researchers [11, 13, 16]. Others contribute to QAP knowledge by presenting a survey of techniques, applications, and methods for the QAP [94, 95].

2.3.3 Methods of Solving. There are many approaches a researcher can try to solve the QAP. But finding an optimal solution is difficult for large problem instances. Usually, only problem instances of size 20 or less can be solved optimally. Moreover, Cela [16] states that problem instances of size 15 are difficult.

In cases where the optimal solution can be found (usually less than size 15), branch and bound methods are typically used [3, 10, 16, 52, 98]. [8] creates a queue based model approach to solve the QAP. [62] uses a bilinear approach to simplify the QAP to a mixed integer-continuous linear program.

Unfortunately, problems larger than size 20 are found in the real-world. These larger problems require the use of stochastic methods to find a good solution in a reasonable time. Below is a list of some of the stochastic methods and algorithms used to solve the QAP.

- Ant Colony Optimization [44, 84, 106]
- Genetic Algorithm [1, 114]
- Evolution Strategies [36, 58, 91, 92]
- Scatter Search Algorithm [22]
- Tabu Search [43]
- Neural Network [61, 107]
- Greedy randomized adaptive search (GRASP) [81]

- Hybrid Methods
  - COSEARCH - Tabu Search & GA [5, 6]
  - Tabu Search, local search & GA operators [37]
  - Genetic algorithm & local search [87–89]
  - Ant colony optimization & local search [111]
  - Simulated annealing & genetic programming [116]
  - GA & Tabu search [121]
  - EA & simulated annealing [126]

The use of parallel methods is another way to gain efficiency and/or effectiveness when solving the QAP. Some of the parallel methods are listed below.

- Branch-and-bound on a computational grid [3]
- Branch-and-bound in parallel using ZRAM [10]
- COSEARCH [5, 6]
- Evolution Strategy using a torus topology [36]
- Genetic Algorithm [55]
- Uses an r-dimensional grid [125]
- Guided Evolutionary Simulated Annealing (GESA) [126]

Many researchers, including most of the ones listed above, use the QAPLIB benchmarks [14] to compare their results with others. Using these problems as benchmarks gives researchers an idea if their method is fruitful or if other methods are better. New benchmark instances are proposed in [32]. These new instances are difficult for meta-heuristic methods to solve.

2.4 Multiobjective Quadratic Assignment Problem (mQAP)

The mQAP is similar to the scalar QAP, with the exception that there are multiple flow matrices – each needing to be minimized. This creates a multiobjective optimization problem (MOP). Section 2.4.1 describes the problem's brief history and defines it in

mathematical terms. Section 2.4.2 discusses the applications that have been applied to the mQAP. Section 2.4.3 looks at the few methods that have been used thus far to solve the mQAP.

2.4.1 History and Definition. The formulation of the mQAP occurred in 2002 [71]. It expands the QAP to include more than one objective. This problem is more feasible to real-world problems that have more than one focus. A factory layout, for example, often has more than one product that needs to be assembled and/or packaged. While the QAP is good for solving one the layout for one product in one location of the factory, the mQAP can take into account all of the products and the entire factory layout. By bringing in more problem knowledge, the researcher can improve the overall efficiency of the layout. The next section lays out the mathematical mQAP definition.

2.4.1.1 Mathematical mQAP definition. Mathematical, the mQAP is defined in equations 2.2 and 2.3

$$minimize\{C(\pi)\} = \{C^1(\pi), C^2(\pi), \ldots, C^m(\pi)\} \tag{2.2}$$

where

$$C^k(\pi) =_{\pi \in P(n)}^{min} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b^k_{\pi_i \pi_j}, k \in 1..m \tag{2.3}$$

and where n is the number of objects/locations, a_{ij} is the distance between location i and location j, b^k_{ij} is the kth flow from object i to object j, π_i gives the location of object i in permutation $\pi \in P(n)$, and 'minimize' means to obtain the Pareto front [72].

In order to determine whether a solution is good or not, metrics are used. Knowles and Corne [72] identified two metrics for use with the mQAP: diameter and entropy. Diameter of the population is defined by Bachelet [4] and is shown in Equation 2.4:

$$dmm(P) = \frac{\sum_{\pi \in P} \sum_{\mu \in P} dist(\pi, \mu)}{|P|^2} \tag{2.4}$$

where $dist(\pi, \mu)$ is a distance measurement that measures the smallest number of two-swaps that need to be performed in order to transform one solution, π, into another solution, μ. The distance measure has a range of $[0, n-1]$.

The metric entropy measures the dispersion of the solutions. It is shown in equation 2.5:

$$ent(P) = \frac{-1}{nlogn} \sum_{i=1}^{n} \sum_{j=1}^{n} (\frac{n_{ij}}{|P|} log \frac{n_{ij}}{|P|}) \tag{2.5}$$

where n_{ij} is a measure of the number of times object i is assigned to the j location in the population.

2.4.2 Applications. Since the mQAP is new, few implementations have been proposed. One proposal applies the mQAP to a group of heterogenous UAVs, where location is the position in a flight formation and the flows are the multiple communication channels between UAVs [25]. Another implementation models communication flow in a constellation of satellites [66]. It has also been applied to the facility layout problem, where there are flows of more than one type of agent [72]. The hospital layout problem is an example of this. Where multiple flows include doctors, nurses, patients, visitors, etc.

2.4.3 Methods of Solving. There are currently three methods used in solving the mQAP. An enumerated search can obtain the optimal solution when the number of locations and facilities are small (around 15 or less) [25, 71, 72]. Larger instances [71, 72] use local search. MOMGA-II, an MOEA based off of the fmGA, is used to solve both large and small instances of the mQAP [25, 66].

2.5 Summary

The QAP is a valuable tool to researchers. Many different problems have been modelled using the QAP. Because of the difficulty of the problem, researchers use innovative methods and approaches in an attempt to find better ways of solving the QAP. But the QAP is limited to only one flow per facility. The mQAP extends the QAP to allow multiple

flows. While this problem is in its infancy, there are many valuable real-world problems that can be mapped to it. And knowing the successes of various algorithms used on the QAP, the researchers can use that knowledge to create good algorithms for the mQAP.

The next chapter presents the background information on multiobjective Evolutionary Algorithms. These stochastic algorithms have been shown to be very effective at solving NP-Complete problems.

3. Multiobjective Evolutionary Algorithm Introduction

3.1 Introduction

Many real-world engineering design problems involve solving multiple objectives simultaneously. A multiobjective optimization problem (MOP) differs from a single-objective optimization problem because it contains several objectives that require optimization versus one. When optimizing a single-objective problem, the best design solution is the goal. But for multiobjective problems, with possibly conflicting objectives, there is usually no single optimal solution. Therefore, the decision maker is required to select a solution by making compromises. A suitable solution should provide for acceptable performance for all objectives.

The operational research field developed many important MOP techniques over the years [19]. Recently, researchers started to apply evolutionary algorithms and concepts in this area. In fact, in the last twenty years, researchers have written more than 1480 papers regarding this topic, many of them occurring in the last five years [17].

This chapter briefly outlines what a multiobjective optimization problem is in section 3.2. Section 3.3.1 goes into more detail about multiobjective Evolutionary Algorithms (MOEAs). The section discusses approaches, theories, metrics, applications, and parallelization techniques.

3.2 Multiobjective Optimization Problems

A MOP consists of decision variables, two or more objective functions, and constraints. Standard MOP and MOEA definitions and nomenclature can be found in [20]. Sections 3.2.1 and 3.2.2 present a few of the more important definitions.

3.2.1 MOP Definitions. The process of finding the global maximum or minimum of any function is referred to as Global Optimization. In general, this is presented in Definition 2 as stated in Bäck [7]:

Definition 2 (Global Minimum): *Given a function $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$, $\Omega \neq \emptyset$, for $\vec{x} \in \Omega$ the value $f^* \triangleq f(\vec{x}^*) > -\infty$ is called a global minimum if and only if*

$$\forall \vec{x} \in \Omega : \quad f(\vec{x}^*) \leq f(\vec{x}) \; . \tag{3.1}$$

when, $\vec{x}^*$ is the global minimum solution(s), f is the objective function, and the set Ω is the feasible region. The problem of determining the global minimum solution(s) is called the global optimization problem. □

This formulation requires some modification in order to reflect the nature of multiobjective problems where there may not be one unique solution but a set of solutions. Multiobjective problems often force the decision maker to make a tradeoff of one solution over another in the objective space.

Multiobjective problems have the task to optimize n objective functions simultaneously. This could include the maximization of all n functions, the minimization of all n functions or a combination of maximization and minimization of these n functions. A MOP and a MOP global minimum (or maximum) is formally defined by Van Veldhuizen as [117]:

Definition 3 (General MOP): *In general, a MOP minimizes (or maximizes) $F(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x}))$ subject to $g_i(\vec{x}) \leq 0, \; i = 1, \ldots, m, \; \vec{x} \in \Omega$. A MOP solution minimizes the components of a vector $F(\vec{x})$ where $\vec{x}$ is a n-dimensional decision variable vector ($\vec{x} = x_1, \ldots, x_n$) from some universe Ω.* □

Definition 4 (MOP Global Minimum): *Given a function $F : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^k$, $\Omega \neq \emptyset$, $k \geq 2$, for $\vec{x} \in \Omega$ the set $\mathcal{PF}^* \triangleq F(\vec{x}_i^*) > (-\infty, \ldots, -\infty)$ is called the global minimum if and only if*

$$\forall \vec{x} \in \Omega : \quad F(\vec{x}_i^*) \preceq F(\vec{x}) \; . \tag{3.2}$$

where, $\vec{x}_i^*,\ i = 1, \ldots, n$ is the global minimum solution set (i.e., $\mathcal{P}^*$), F is the multiple objective function, and the set Ω is the feasible region. The problem of determining the global minimum solution set is called the MOP global optimization problem. □

This MOP consists of k objectives represented by the k objective functions, m constraints on the objective functions and n decision variables. The k objective functions are either linear or nonlinear in nature. The evaluation function, $F : \Omega \longrightarrow \Lambda$, is a mapping from the decision variables ($\vec{x} = x_1, \ldots, x_n$) to output vectors ($\vec{y} = a_1, \ldots, a_k$) [117].

Some additional terminology is necessary in order to remain consistent with the terminology used in the field. The ultimate goal of the MOP is called the *objective* and the coordinate space where plots of vectors resulting from the MOP evaluation is called the *objective space* [117].

3.2.2 Pareto Terminology. The concept of Pareto Optimality is vital to the theory and analysis of MOPs. It enables the researcher to determine if one solution is "better" than another. In addition, these Pareto concepts allow for the determination of a set of optimal solutions in MOPs. Some of the key Pareto concepts, for minimization MOPs, are defined mathematically by Van Veldhuizen as [117]:

Definition 5 (Pareto Dominance): *A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to dominate another vector $\vec{v} = (v_1, \ldots, v_k)$ (denoted by $\vec{u} \preceq \vec{v}$) if and only if u is partially less than v, i.e., $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$.* □

Definition 6 (Pareto Optimality): *A solution $x \in \Omega$ is said to be Pareto optimal with respect to Ω if and only if there is no $x' \in \Omega$ for which $\vec{v} = F(x') = (f_1(x'), \ldots, f_k(x'))$ dominates $\vec{u} = F(x) = (f_1(x), \ldots, f_k(x))$. The phrase "Pareto optimal" is taken to mean with respect to the entire decision variable space unless otherwise specified.* □

Definition 7 (Pareto Optimal Set): *For a given MOP $F(x)$, the Pareto optimal set ($\mathcal{P}^*$) is defined as:*

$$\mathcal{P}^* := \{x \in \Omega \mid \neg\exists\, x' \in \Omega \;\; F(x') \preceq F(x)\}. \tag{3.3}$$

□

Definition 8 (Pareto Front): *For a given MOP $F(x)$ and Pareto optimal set $\mathcal{P}^*$, the Pareto front ($\mathcal{PF}^*$) is defined as:*

$$\mathcal{PF}^* := \{\vec{u} = F(x) = (f_1(x), \ldots, f_k(x)) \mid x \in \mathcal{P}^*\}. \tag{3.4}$$

□

The Pareto optimal solutions are ones within the search space whose corresponding objective vector components cannot be improved simultaneously. These solutions are also known as *non-inferior*, *admissible*, or *efficient* solutions, with the entire set represented by $\mathcal{P}^*$. Their corresponding vectors are known as *nondominated*; selecting a vector(s) from this vector set (the Pareto Front set $\mathcal{PF}^*$) implicitly indicates acceptable Pareto optimal solutions (**genotypes**). These are the set of all solutions whose vectors are nondominated; these solutions are classified based on their *phenotypical* expression. Their expression (the nondominated vectors), when plotted in criterion (**phenotype**) space, is known as the *Pareto front* [117, 135].

Armed with the basic MOP definitions, we are now ready to delve into the specifics of MOEAs.

3.3 Multiobjective Evolutionary Algorithms

This section gives background information about MOEAs. For more in depth discussion on MOEAs, the book by Coello, et. al. [20] is an excellent source. Other good sources for MOEA information include [18, 19, 28, 41, 127]

3.3.1 MOEA Approaches. MOEA approaches are generally categorized into three groups, based on the techniques that they use. These categories are as follows [20]:

- *A priori* Techniques
- *Progressive* Techniques
- *A posteriori* Techniques

A more in depth discussion of these three categories and the techniques they include continues in the next few sections.

3.3.1.1 A priori *Techniques.* These techniques require the researcher to weight the objectives before the search process begins. But knowing what value to weight each objective is a difficult task. Bad weighting choices can prevent the researcher from implementing a better solution.

There are three techniques that fit into this category. They include the following [20]:

- Lexicographic
- Linear fitness combination
- Non-linear fitness combination

Lexicographic ordering. This technique requires the user to rank the objectives in order of importance. The algorithm then minimizes the objective functions based on the priority given by the user. This approach is seldom used, possibly due to the unequal searching of the objective space.[20]

Linear fitness combination. This technique generally computes the fitness using a form similar to equation :

$$fitness = \sum_{i=1}^{k} w_i f_i(\vec{x}) \tag{3.5}$$

where the weighting coefficients, $w_i \geq 0$, determine the relative importance of each objective. It is commonly assumed that the sum of all the weights is equal to one. This approach

is popular, possibly due to its simplicity. But it turns out that if PF_{true} is non-convex, then the optima cannot be found in that region [20]. An example of this technique can be seen in [120].

Non-linear fitness combination. This technique usually involves a multiplication of the objective functions. This technique is rarely used, possibly due to the amount of overhead required to come to a solution.

Since these techniques limit the exploration, they cannot find all $Pareto_{true}$ values.

3.3.1.2 Progressive *Techniques.* These techniques require interaction between the researcher and the algorithm. This requires the researcher be involved with the program while it is running to guide it toward better solutions. This also requires the researcher to define goals in order to bias the search. As such, this method is rarely used [20]. The MOGA is an example of this technique put into practice [39, 40]

3.3.1.3 A posteriori *Techniques.* These techniques are the most common in literature. They explicitly seek P_{true}. There are five techniques that fit into this category. They include the following [20]:

- Independent sampling
- Criterion selection
- Aggregation selection
- Pareto-based selection
- Hybrid selection

Independent sampling. This technique involves some fitness combination where the weights of the objective are varied between a number of separate runs. This technique is simple and efficient. But its usefulness is limited because the arbitrary weight combinations possibly limit finding some $Pareto_{true}$ values. [20]

Criterion selection. This technique divides each generation into subpopulations, based on how many objective values require solving. Each of these subpopulations bases its

selection of individuals for the next generation on one objective function. Once each subpopulation selects its new generation, all of the subpopulations are combined into one large population, the same size as the original. The individuals are shuffled in the population and the genetic operations are applied to each. The process repeats until the stopping criteria is reached. A disadvantage of this technique is a problem called "speciation". Speciation occurs because population members are selected based on one dimension of performance. This type of selection favors individuals located on the outer edges of $PF_{current}$ and leaves individuals located in the center at a disadvantage for selection. One method to overcome this problem requires a heuristic that gives selection preference to nondominated individuals in each generation [20]. The Vector Evaluated Genetic Algorithm (VEGA) is an example of this type of program [102–104].

Aggregation selection. These techniques involve weighting various aspects of the problem and getting their aggregate values. The weights are applied in different ways and to different aspects of the problem. Using weighted sums has the disadvantage that some individuals on PF_{true} may be missed [23]. The Multi-Objective Genetic Local Search Algorithm is an example of this technique [60].

Pareto-based selection. These techniques use Pareto-based fitness assignments to find nondominated members on $PF_{current}$. There are many versions of this approach. Since this thesis uses a Pareto-based approach, the differing approaches are outlined in section 3.3.1.4. The biggest disadvantage of using this type of approach is the lack of an efficient algorithm to find the nondominated members. The conventional process for finding the nondominated points in a generation has a complexity of $O(kM^2)$ where k is the number of objectives and M is the size of the population [20]. The Thermodynamical Genetic Algorithm is an example of this technique [65].

Hybrid selection. This technique uses multiple selection mechanisms instead of just one. Generally, a hybrid selection method tries to combine the selection mechanism from several good MOEAs and apply each selection method at some point during the algorithms execution. This approach attempts to mitigate any shortcomings that one selection method may have by combining it with a complimentary selection method. [20].

3.3.1.4 Pareto-based Approaches. This section discusses some of the various approaches researchers attempt when they use the Pareto-based approach. This section on gives cursory look at these algorithms. For more information, an excellent description of these is found in [20].

Goldberg's Pareto Ranking Goldberg suggested moving the population toward PF_{true} by using Pareto nondominated points and selection [46]. To get the population for the next generation, the nondominated Pareto Fronts are determined and are ranked based on best solution set to the worst. Once the number of individuals ranked matches the number of individuals needed for the next generation, the process is terminated.

Multi-objective Genetic Algorithm (MOGA) Fonseca uses a ranking approach different from Goldberg. He ranks the points based on how many other points dominate them. His first rank is identical to Goldberg's first. But the rest of the ranks are dependent upon how dense the population is in front of the point. MOGA uses a niche-formation method in order to diversify the population [39]. Since MOGA niching is done in the objective space, individuals that map to the same objective value will only have one member kept in the population and all others removed. This is a disadvantage of the algorithm [20].

Nondominated Sorting Genetic Algorithm (NSGA) This method, presented in [109] ranks members based on the size of the population when they are nondominated. This results in the better members getting the higher fitness scores. Selection is done using stochastic remainder proportionate selection to ensure copies are distributed properly. An offshoot of this approach, NSGA-II, is more efficient and uses elitism. This method tends to perform worse than MOGA in tests, this may be due to the sharing factor being improperly set [20].

Niched-Pareto Genetic Algorithm (NPGA) This method employs an interesting form of tournament selection called Pareto domination tournaments. Two members of the population are chosen at random and they are each compared to a subset of the population. If one is nondominated and the other is not, then the nondominated one is

selected. If there is a tie (both are either dominated or nondominated), then fitness sharing decides the tourney results [56].

Strength Pareto Evolutionary Algorithm (SPEA) This method attempts to integrate different MOEAs. First introduced in [132], the algorithm uses a strength variable that is similar to the MOGA ranking system. Each member of the population is assigned a fitness value according to the strengths of all nondominated solutions that dominate it. Diversity is maintained through the use of a clustering technique called the "average linkage method".

A revision of this method, called SPEA2 [129], adjusts slightly the fitness strategy and uses nearest neighbor techniques for clustering. In addition, archiving mechanism enhancements allow for the preservation of boundary solutions that are missed with SPEA.

Multi-Objective Messy Genetic Algorithm (MOMGA) This method extends the mGA [29] to solve multiobjective problems. The MOMGA [117] is an explicit building block GA that produces all building blocks of a user specified size. The algorithm has three phases: Initialization, Primordial, and Juxtapositional. For explicit details of how this algorithm functions, see Section 4.3. A disadvantage of this algorithm is the exponential growth of the population as the building block size grows.

Multiobjective Hierarchical Bayesian Optimization Algorithm (mhBOA) This explicit building block method combines the multiobjective selection scheme of NSGA-II with the hBOA [96]. The mhBOA [64] is a linkage learning algorithm that attempts to define tight and loose linkages to building blocks in the chromosome. This method uses a Bayesian network (a conditional probabilistic model) to guide the search toward a solution. A disadvantage of this algorithm is the time it takes to generate results with just a small number of linkages tested.

Pareto Archived Evolution Strategy (PAES) This method, formulated by Knowles and Corne[69], uses a (1+1) evolution strategy, where each parent generates one offspring. The method uses an archive of nondominated solutions to compare with individuals in the current population. For diversity, the algorithm generates a grid overlaid on

the search space and counts the number of solutions in each grid. A disadvantage of this method is its performance on disconected Pareto Fronts.

General Multi-Objective Program (GENMOP) This method is a parallel, real-valued MOEA initially used for bioremediation research [67]. This method archives all previous population members and ranks them. Archived individuals with the highest ranks are used as a mating pool to mate with the current generation. The method uses equivalence class sharing for niching to allow for diversity in the mating pool. A disadvantage of this algorithm is the Pareto ranking of the archived individuals at each generation.

Pareto-based selection These are approaches that do not use niching, sharing, or crowding. In order to maintain diversity, other methods need to be devised. Several different approaches are described in [20].

Pareto deme-gased selection This approach applies Pareto ranking on many small subpopulations. This approach fits nicely into the parallel processing paradigm. A new method must be created in order to determine which nondominated subpopulation members are also globally nondominated.

Pareto elitist-based selection These approaches take the best n individuals from one generation and propagate them to the next. After that the rest of the population is filled using some other method. Large selection pressure for this approach can cause premature convergence [20].

3.3.2 Metrics. Metrics are important in any field. They allow the researcher to gauge the performance of his algorithm. The MOEA field is no different. MOEA metrics tend to focus on the phenotype or objective domain. This is contrary to what most operations researchers do. They tend to use metrics in the genotype domain. But since there is an explicit mapping between the two, it doesn't matter in which domain you obtain your metrics [20, 119].

MOEA metrics can be used to measure final performance or track the generational performance of the algorithm. This is important because it allows the researcher to look how the algorithm converges during execution. If he notices that the algorithm converges

prematurely, then he can adjust his parameter settings in order to slow down the convergence and allow for more exploration. This section looks at some of metrics used in the study of MOEAs.

Error Ratio (ER): This metric reports the number of vectors in PF_{known} that are not members of PF_{true}. This metric requires that the researcher knows PF_{true}. The mathematical representation of this metric is shown in equation 3.6:

$$ER \triangleq \frac{\sum_{i=1}^{n} e_i}{n} \tag{3.6}$$

where n is the number of vectors in PF_{known} and e_i is a zero when the i vector is an element of PF_{true} or a 1 if i is not an element. [20]

So when $ER = 0$, the PF_{known} is the same as PF_{true}; but when $ER = 1$, this indicates that none of the points in PF_{known} are in PF_{true}.

Two Set Coverage (CS): This metric is named in [20], but was originally defined in [128]. This metric compares the coverage of two competing sets and outputs the percentage of individuals in one set dominated by the individuals of the other set. This metric does not require that the researcher have knowledge of PF_{true}. The equation for this metric is shown in equation 3.7:

$$CS(X', X'') \triangleq \frac{|a'' \in X''; \forall a' \in X' : a' \succeq a''|}{|X''|} \tag{3.7}$$

where $X', X'' \subseteq X$ are two sets of phenotype decision vectors, and (X', X'') are mapped to the interval $[0, 1]$. This means that $CS = 1$ when X' dominates or equals X''.

Zitzler uses this a metric in several publications [130–132].

Generational Distance (GD): This metric is defined in [20, 118]. It reports how far, on average, PF_{known} is from PF_{true}. This metric requires that the researcher knows PF_{true}. It is mathematically defined in equation

$$GD \triangleq \frac{(\sum_{i=1}^{n} d_i^p)^{1/p}}{n} \tag{3.8}$$

where n is the number of vectors in PF_{known}, $p = 2$, and D_i is the Euclidean distance between each member and the closest member of PF_{true}, in the phenotype space. When $GD = 0$, $PF_{known} = PF_{true}$.

Hyperarea and Ratio (H,HR): These metrics, discussed in [20, 131], define the area of coverage that PF_{known} has with respect to the objective space. This would equate to the summation of all the areas of rectangles, bounded by the origin and $(f_1(\vec{x}), f_2(\vec{x}))$, for a two-objective MOEA. Mathematically, this is described in equation 3.9:

$$H \stackrel{\triangle}{=} \left\{ \bigcup_i a_i | v_i \in PF_{known} \right\} \tag{3.9}$$

where v_i is a nondominated vector in PF_{known} and a_i is the area of the calculated between the origin and vector v_i. But if PF_{known} is not convex, the results can be misleading. It is also assumed in this model that the origin is $(0, 0)$

The hyperarea ratio metric definition can be seen in equation 3.10:

$$HR \stackrel{\triangle}{=} \frac{H_1}{H_2} \tag{3.10}$$

where H_1 is the PF_{known} hyperarea and H_2 is the hyperarea of PF_{true}. This results in $HR \geq 1$ for minimization problems and $HR \leq 1$ for maximization problems. For either type of problem, $PF_{known} = PF_{true}$ when $HR = 1$. This metric requires that the researcher knows PF_{true}.

Spacing (S): This metric outputs the spread of the vectors in PF_{known}. Coello describes this metric from [105] in his book [20]. This metric measures the distance variance of neighboring vectors in PF_{known}. Equation 3.11 defines this metric.

$$S \stackrel{\triangle}{=} \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\bar{d} - d_i)^2} \tag{3.11}$$

and

$$d_i = min_j(|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|) \tag{3.12}$$

where $i, j = 1 \ldots, n$, $\bar{d}$ is the mean of all d_i, and n is the number of vectors in PF_{known}. When $S = 0$, all members are spaced evenly apart. This metric does not require the researcher to know PF_{true}.

Overall Nondominated Vector Generation Ratio (ONVGR): This metric measures the total number of nondominated vectors during MOEA execution and divides it by the number of vectors found in PF_{true}. Coello [20] defines this metric as shown in equation 3.13:

$$ONVG \triangleq \frac{PF_{false}}{PF_{true}} \tag{3.13}$$

When $ONVGR = 1$ this states only that the same number of points have been found in both PF_{true} and PF_{known}. It does not infer that $PF_{true} = PF_{known}$. This metric requires that the researcher knows PF_{true}.

Progress Measure RP: For single-objective EAs, Bäch [7] defines a metric that measures convergence velocity. This single-objective metric is applied to multiobjective MOEAs in [20], and is shown in equation 3.14:

$$RP \triangleq ln\sqrt{\frac{G_1}{G_T}} \tag{3.14}$$

where G_1 is the generational distance for the first generation and G_T is the distance for generation T. Recall that generational distance was defined in equation 3.8 and it measures the average distance from PF_{true} to PF_{known}. This metric requires that the researcher knows PF_{true}.

Generational Nondominated Vector Generation (GNVG): This is a simple metric, defined in [20] that lists the number of nondominated vectors produced for each generation. This is defined in equation 3.15

$$GNVG \triangleq |PF_{current}(t)| \tag{3.15}$$

This metric does not require the researcher know PF_{true}.

Nondominated Vector Addition (NVA): This metric, defined in [20], calculates the number of nondominated vectors gained or lost from the previous PF_{known} generation. Equation 3.16 defines this metric.

$$NVA \triangleq |PF_{known}(t)| - |PF_{known}(t-1)| \tag{3.16}$$

But this metric can be misleading when a new vector dominates two or more vectors from the previous generation. In addition, this metric may remain static over the course of several generations while new points are added that dominate others from the previous generation. This metric does not require the researcher know PF_{true}.

Table 3.1 lists the various MOEA metrics and states whether they require PF_{true} or explicitly compare results from one generation to another.

Table 3.1 **Summary of MOEA Metrics**

Metric Name	PF_{true} required?	Generational Metric?
Error Ratio	Yes	No
Two Set coverage	No	No
Generational Distance	Yes	Yes
Hyperarea	No	No
Hyperarea Ratio	Yes	No
Spacing	No	No
ONVGR	Yes	Yes
Progress Measure	Yes	Yes
GNVG	No	Yes
Nondominated Vector Addition	No	Yes

3.3.3 MOEA Applications. MOEAs have been used on many real world applications. For a nearly exhaustive list of applications, see either the book by Coello [20] or his website which lists nearly all of the MOEA research done in the field. The list below has just a sampling of some of the recent research done in MOEAs.

- Base station transmitter placement for a cellular network [122]
- Wire antenna geometries [15, 120, 123]
- Protein Structure Prediction Problem [24, 26, 27]
- Groundwater pollution remediation [34, 56, 67]
- VLSI placement [99, 100]
- Network design [78, 112, 113]
- Offline Routing [73–75]
- Robot path planning [31]

3.4 Summary

This chapter provides an overview on MOPs. Discussion continues regarding MOEAs and the many methods that have been undertaken. MOEA metrics are presented after that. Several MOEA applications are presented in an effort to show the broad range of real world problems that MOEAs are used to solve.

The next chapter discusses in more detail an MOEA called MOMGA-II. This algorithm is the one selected to use on the problem.

4. MOMGA-II

4.1 Introduction

This chapter describes the Multiobjective Messy Genetic Algorithm - II (MOMGA-II) algorithm. The majority of the experiments in this research use this algorithm. The MOMGA-II algorithm contains many modifiable parameters. It can be run in serial or parallel mode. It can be run as a Messy Genetic Algorithm (mGA) or as a Fast Messy Genetic Algorithm (fmGA). Many of the parameters can be set or turned off. Section 4.2 describes fmGA, which is one of the two algorithms that MOMGA-II is modelled after. The fmGA inherited much of its structure from its predecessor, the mGA. For more information on this algorithm, see Appendix C. Section 4.3 discusses the MOMGA program, the predecessor of MOMGA-II. Section 4.4 discusses the MOMGA-II program.

4.2 Fast Messy Genetic Algorithms

The fmGA was designed in 1993 by Goldberg, Deb, Kargupta, and Harik [48]. It was designed as an attempt to overcome some of the previously mentioned flaws in the mGA. For further information on the mGA, see Appendix C. The fmGA replaced the PEI portion of the mGA with probabilistically complete initialization (PCI). Appendix C.5.1 describes the PEI and other initialization features of the mGA. And the primordial phase was also replaced by a building-block filtering phase. The next few sections briefly describe PCI and building-block filtering and their advantages over the previous methods used in the mGA.

4.2.1 Probabilistically Complete Initialization. One of the biggest problems of the mGA was its PEI implementation. This method of generating building blocks proved to be unmanageable for anything other then pedagogical examples. In order to create a more effective algorithm, the fmGA implements a new initialization technique, PCI. PCI differs from PEI in that it creates a **controlled number of copies** of building blocks of a specified size, whereas the PEI generates **all** of the building blocks of a specified size.

According to Goldberg [48], there are two important factors in the initialization process: initial string length and population size. For the fmGA to be effective, the

initialization phase must be able to create a big enough population and a long enough chromosome that can handle all possible genic and allelic combinations [68]. To determine the proper initial string length and population size, PCI is used.

The premise behind PCI is that all of the $2^k\binom{l}{k}$ equivalence classes can be defined by using smaller amount of strings when the string length is much longer than k [63]. Basically, this means that one string can be used to define multiple classes.

To determine the proper sizing of the initial string and population, we first need to set up the formula. First, let the initial string length be l', let l be the number of genes, and let k be the size of the gene combination. Also let $l' < l$ and $l' > k$. The total number of strings created of length l' with l genes is $\binom{l}{l'}$. And the total number of ways a string of size l' contains gene combinations of size k is calculated by assigning k genes to the l' sized string and then choosing the ways that $l' - k$ genes can be created using $l - k$ genes. This gives the formula of $\binom{l-k}{l'-k}$ [48]. Putting these two formulas together, gives us equation 4.1, which is the probability of randomly selecting a k sized gene combination in a l' length string with a total of l genes to choose from.

$$p(l', k, l) = \frac{\binom{l-k}{l'-k}}{\binom{l}{l'}} \tag{4.1}$$

The inverse of this equation can be found in equation 4.2. This equation suggests that random n_g string samples of length l' will produce, on average, one string that will have the desired gene combination of size k [48]. It can be shown that as the initial string length l' increases, the population size, n_g decreases. Figure 4.1 depicts how n_g varies with respect to l' and k. Note that when $l' \approx l$, the value for n_g is constant and doesn't depend on l. It can also be shown that for large values of both l and l', $n_g \approx (\frac{l}{l'})^k$.

$$n_g = \frac{\binom{l}{l'}}{\binom{l-k}{l'-k}} \tag{4.2}$$

In order to include all the allele combinations up to order k, the n_g will have to be multiplied to a factor that accounts for all 2^k allele combinations and the noise involved in building-block evaluations. The population sizing equation chosen by Goldberg [48] was

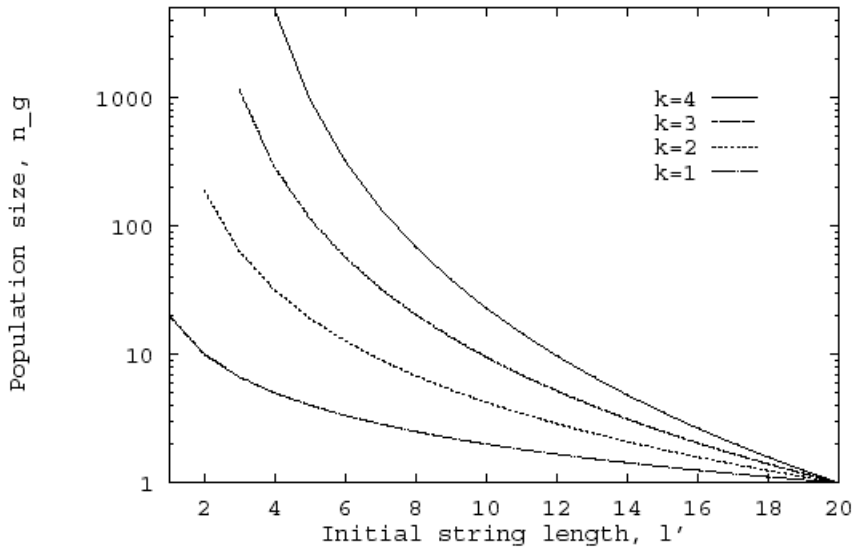

Figure 4.1 The population size required to have one expected instance of a building block of size k in strings of length l' is plotted against l'. The problem size is assumed to be $l = 20$ [48]

taken from earlier work he did on population sizing [47]. The n_a equation, equation 4.3 was developed for simple GAs and it was designed to account for the noise in building block evaluation.

$$n_a = 2c(\alpha)\beta^2(m-1)2^k \tag{4.3}$$

The population size is calculated in such a way that the selection error between two building-blocks is no more than α. The parameter $c(\alpha)$ is the square of the ordinate of a normal random deviate whose tail has an area (error probability) of α, the number of subfunctions is denoted by m, and β^2 is the maximum signal-to-noise ratio, which is the ratio of the variance and the mean of the convolution of the distribution of the two competing classes [47, 48, 63].

Now that we have n_g and n_a, we can multiply them together to get the overall population size, n. This equation is shown in equation 4.4.

$$n = n_g n_a = \frac{\binom{l}{l'}}{\binom{l-k}{l'-k}} 2c(\alpha)\beta^2(m-1)2^k \tag{4.4}$$

In order to find a good initial string length, l', there is a trade-off that must be decided upon. A large l' will decrease the n_g value, but it will increase the error probability, because of noise introduced with the additional bits of the string length. Goldberg [48] and Kargupta [63] both decided to use $l' = l - k$. This equates to the overall population sizing of $O(l)$, which is a great improvement over the mGA initialization phase.

4.2.2 Building-block Filtering. In order for the fmGA to function properly, the initial string length needs to be reduced to a size near the building-block length, k. Building-block filtering is the process of doping the population with good building-blocks and then randomly deleting genes from the strings. This section will describe the building-block filtering process that was implemented by Goldberg [48]

To explain this concept, we first need to introduce some symbols. Start by considering a sequence of string lengths created by successive applications of gene deletion, $\lambda^{(0)}, \lambda^{(1)}, ...\lambda^{(i)}, ...\lambda^{(N)}$, where $\lambda^{(0)} = l'$ and $\lambda^{(N)} \approx k$. The ith length reduction can be defined as $\rho_i = \lambda^{(i)}/\lambda^{(i-1)}$. So at the ith stage, the number of genes randomly deleted from each string is $\lambda^{(i-1)} - \lambda^{(i)}$. Once the reduction is complete, selection is done for a specified number of generations in order to increase the number of good building-blocks. Then deletion occurs and the process continues until $\lambda^{(N)}$ is reached.

The rate of deletion needs to be selected in manner so that it is, on average, less than rate of doping. To correctly choose a building-block of size k from strings of size $\lambda^{(i-1)}$ by picking $\lambda^{(i)}$ genes at random, we need a building-block repetition factor $\gamma = \binom{\lambda^{(i-1)}}{\lambda^{(i)}}/\binom{\lambda^{(i-1)}-k}{\lambda^{(i)}-k}$ strings to have one expected copy remaining of the desired building-block [48]. For large values of $\lambda^{(i-1)}$ and $\lambda^{(i)}$ compared to k, γ varies as $(\lambda^{(i-1)}/\lambda^{(i)})^k$. We may choose $\lambda^{(i)}$ so that γ is smaller than the number of duplicates generated by the selection operator. One method of designing a gene deletion schedule is to set ρ to a constant value much smaller than 2^{t_s}, where t_s is the total number of selections repetitions per length reduction. Goldberg did this because he expected binary tournament selection

```
Initialize population with random strings of size l'
While (l' > ζk)
{
    Do successive selections with no evaluations
    Choose l''(< l') genes in a random fashion
    Set l' = l''
    Evaluate the new strings
}
```

Figure 4.2 Pseudo-code depicting fmGA PCI and building-block filtering interaction

to approximately double the proportion of the best individuals during each invocation [48]. Using the asymptotic relation for $\gamma = (\lambda^{(i-1)}/\lambda^{(i)})^k = \rho_i^{-k}$, the assumed fixed γ roughly implies a fixed length-reduction ratio $\rho = \rho_i$, for all i, which the total number of length reductions required to reduce the string length to $O(k)$. If it is assumed that the final string length is equal to ζk, where $\zeta \geq 1$, the number of length reductions (t_r) required is shown in equation 4.5.

$$\frac{l'}{\rho^{t_r}} = \zeta k \tag{4.5}$$

This simplifies to become equation 4.6.

$$t_r = \frac{log(l/\zeta k)}{\rho} = \zeta k \tag{4.6}$$

This equation shows that if the $\lambda^{(i)}$ values are chosen to make the deletion so that the length-reduction factor ρ is a constant, then t_r varies as $O(\log l)$. And with a population size of $O(l)$, combined with the fact that the evaluation of strings is performed only once after each length reduction, the overall complexity of PCI and building-block filtering is expected to be $O(l\log l)$. Figure 4.2 shows pseudo-code that depicts how PCI and building-block filtering work.

According to Kargupta [63], cross-competition among the different building blocks can lead to uneven growth. He recommends further modifications that should enhance the performance of the fmGA, including an alternate approach to the building-block filtering

described previously. These modifications won't be discussed here, but the reader can read about these more in depth in the paper written by Kargupta [63].

4.3 MOMGA

(MOMGA) stands for "Multiobjective mGA". It is a program written by David Van Veldhuizen and discussed in his dissertation [117]. This program is briefly discussed here as it is the starting point for the MOMGA-II code. This section looks at the implementation of the algorithm and shows the baseline that the MOMGA-II started from.

4.3.1 Fitness Functions. MOMGA uses fitness functions that operate over the entire l-bit string. It does not use subfunctions which operate on only a part of the string. The reason for following this path is two-fold. It focuses only on the building-blocks used in MOP solutions. It also helps to prevent problems in determining relationships (if any), between subfunctions and a complete MOP. [117]

4.3.2 Solution Evaluations. When compared to the mGA, the MOMGA has a similar population size. But the MOMGA's algorithmic complexity is different than the mGA due to the multiple function evaluations that must be run. Table 4.1 shows a comparison of the two complexities.

Table 4.1 **Comparison of MOMGA and mGA Complexity**[117]

Algorithm	Number of objectives	Complexity
mGA	Single objective	$O(2^k\binom{l}{k}+C)$
MOMGA	p objectives	$O(p2^k\binom{l}{k}+pC)$

While the equation only shows a linear increase in the complexity, some real-world fitness functions are rather time-consuming, so a linear increase adds a significant amount of time to the process.

As far as storage requirements go, the MOMGA stores a vector that contains the values corresponding to the number of functions being optimized. Whereas the mGA stores only a single value, since it is not multi-objective. So while both increase in a linear fashion, the MOMGA will tend to increase at a higher rate than the mGA.

4.3.3 Evolutionary Operators.

4.3.3.1 Selection Operator. The MOMGA, as in mGA, uses a tournament size of 2 in its selection process. This equates to a medium selection pressure [7]. The MOMGA implements a modified, Pareto-based tournament selection operator based on the one implemented in the NPGA algorithm [56]. This modified selection operator randomly selects two individuals for the tournament, but in addition, a comparison set (t_{dom}) of individuals is also chosen. Then, using Pareto dominance, the two individuals are compared with each individual in the comparison set. If one individual is found to be nondominated and the other individual is not, then the nondominated individual is selected. But if neither is dominated or if both are dominated, then sharing is implemented. The goal of sharing is to distribute the population across the peaks of the search space so that each peak receives a fraction of the population in proportion to its height [56]. The form of sharing used in MOMGA is equivalence class sharing. This type of sharing is implemented when there is no clear winner among the competing individuals. In this type of sharing, the individual with the smallest niche count is the one that is chosen.

Horn implemented this selection process because he found that using binary selection alone, he was getting poor domination pressure (selection pressure) which in turn created poor representations of PF_{known}. The comparison set introduced was used to control the *domination pressure*. Horn also reports what values he empirically found to be good values for the dominance pressure in another paper [57].

4.3.3.2 Cut-and-Splice. The MOMGA uses the cut-and-splice operator in the same manner as implemented in the mGA.

4.3.3.3 Mutation. The MOMGA uses the mutation operator in the same manner as defined in the mGA.

4.3.4 Competitive Templates. The mGA uses a competitive template in the primordial stage, where all the building-blocks are evaluated against it, and in the juxtapositional phase, where it evaluates the fitness of the newly created individuals. Each

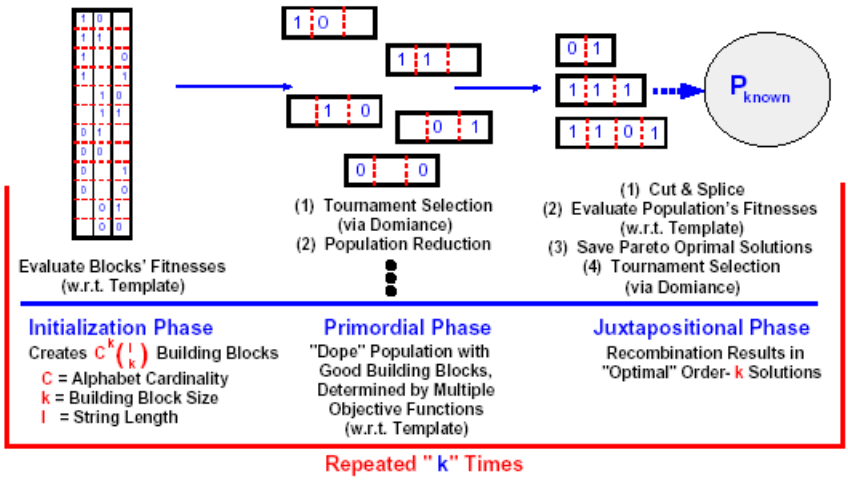

Figure 4.3 MOMGA operation [117]

partial string's assigned fitness is a template fitness where unassigned values are filled with ones from the template. The template is first generated randomly, and after each era, the template is replaced with the most fit example that has been found. So the template is locally optimal to the previous era.

Since the mGA is a single objective problem, only one competitive template is needed to guide the search. But since MOMGA is a MOP, it has a set of solutions and using a template for each solution is not feasible. So the MOMGA employs a different competitive template during the primordial and juxtapositional phases. Each competitive template is associated with each objective function being optimized. This is done by randomly selecting one of the k templates. At the end of each era, the best solutions for each objective are used to replace the previous template for the objective. Van Veldhuizen notes that this approach may result in strong "speciation" [42, 109], but notes that his focus is on determining the use and role of building-blocks in forming MOP solutions [117]

4.3.5 MOMGA Operation. The MOMGA operation was graphically represented by van Veldhuizen in Figure 4.3. In addition, the MOMGA pseudocode is shown in Figure 4.4.

At the end of each era, after the juxtapositional phase is run, the $P_{current}(t)$ is added to $P_{known}(t-1)$. When all eras complete and MOMGA terminates, all of the solutions

```
For n = 1 to k
   Perform Partially Enumerative Initialization
   Evaluate Each Population Member's Fitness (w.r.t k Templates)
// Primordial Phase
   For i = 1 to Maximum Number of Primordial Generations
      Perform Tournament Thresholding Selection
      If (Appropriate Number of Generations Accomplished)
         Then Reduce Population Size
      End If
   End Loop
// Juxtapositional Phase
   For i = 1 to Maximum Number of Juxtapositional Generations
      Cut-and-Splice
      Evaluate Each Population Member's Fitness (w.r.t. k template
      Perform Tournament Thresholding Selection and Fitness Sharing
      P_known(t) = P_current(t) ∪ P_known(t − 1)
   End Loop
Update k Competitive Templates (Using Best Value Known in Each Objective)
End Loop
```

Figure 4.4 MOMGA Pseudo-code [117]

in P_{known} are tested. Every vector that is dominated is removed from P_{known}. Solution culling is performed in order to avoid slowing down the execution of the algorithm [117].

4.4 MOMGA-II

The MOMGA-II algorithm was developed by Jesse Zydallis as part of his dissertation [133]. It was developed as an attempt to expand the state of the art for explicit building-block MOEAs. While there has been a lot of research done for single objective explicit building-block EAs [24, 29, 48, 50, 68, 86, 90], there has been limited research of explicit building-blocks in the MOEA realm [117, 133].

MOMGA-II used MOMGA as its foundation with respect to an explicit building-block MOEA. But in order to improve efficiency, the MOMGA-II moves away from the mGA approach and uses the fmGA approach. The SO fmGA [48] was the template for the fmGA. By combining these two approaches, MOMGA-II takes advantage of the best concepts of both and extends them further.

This section discusses the differences between the MOMGA and MOMGA-II algorithms and many of the features and options written into MOMGA-II.

4.4.1 MOMGA-II vs. MOMGA. MOMGA was based upon the mGA while MOMGA-II used the fmGA as its starting point. It is widely known that the mGA has a bottleneck in the initialization phase because it enumerates every building-block, up to the user specified size. This bottleneck makes any mGA based algorithm infeasible when dealing with problems that have large bit string lengths. Since MOMGA-II uses the fmGA, it avoids the large population sizes generated by the mGA. This make the MOMGA-II more feasible for larger problems, as well as more efficient.

MOMGA-II uses PCI and building-block filtering, which have been described previously in the section on the fmGA. These techniques ensure that the fmGA has the same effectiveness as the mGA, while improving the efficiency. In fact, Zydallis [133] found that MOMGA-II could get similar effectiveness and better efficiency with a population size of 250 than the MOMGA could with a population size of 16,192 for identical MOPs.

MOMGA-II is similar to MOMGA in the way it handles the competitive templates. They are randomly created initially, they are updated at the end of an era, and they are chosen at random when used to fill in bits for under-specified members.

Table 4.2 **Comparison of MOMGA-II and MOMGA**

	MOMGA-II	MOMGA
Algorithm used as baseline	fmGA	mGA
Initialization technique used	PCI	PEI
Phases of algorithm	Initialization Building-block Filtering Juxtapostional	Initialization Primordial Juxtapositional
Competitive template selection	Random	Random

MOMGA-II operates the same as MOMGA in the Juxtapositional phase and the bookkeeping that occurs after the juxtapositional phase (i.e. building-block size is incremented, competitive templates replaced if necessary, and PF_{known} is generated in the same fashion). Table 4.2 show a comparison of some of the distinguishing features.

To better comprehend how the MOMGA-II program operates, the psuedo-code is presented. Figure 4.5 shows the MOMGA-II pseudo-code.

Figure 4.6 shows the flow of the MOMGA-II program. The dashed boxes show what functions belong to various parts of the program.

4.5 Parallelization of MOMGA-II

The MOMGA-II program has several parallel options. For our experiments, when parallelization was used, we used the island model. See Figure 4.7 for a graphical depiction of the island model. For more information on the island model and other parallel techniques, see Appendix B. The island model was chosen because it runs with multiple populations at a time while the diffusion and master-slave model run only one population at a time. This speeds up the process, because the Pareto analysis is the slowest part of the algorithm and if that can be done at the same time over multiple machines, it helps to increase the efficiency.

```
For n = 1 to o
   Perform Partially Complete Initialization
   Evaluate Each Population Member's Fitness (w.r.t k Templates)
// Building Block Filtering Phase
   For i = 1 to Maximum Number of Building Block Filtering Generations
      If(Building Block Filtering Required Based Off of Input Schedule)
         Then Perform Building Block Filtering
      Else
         Perform Tournament Thresholding Selection
      End If
   End Loop
// Juxtapositional Phase
   For i = 1 to Maximum Number of Juxtapositional Generations
      Cut-and-Splice
      Evaluate Each Population Member's Fitness (w.r.t. k template
      Perform Tournament Thresholding Selection and Fitness Sharing
      P_known(t) = P_current(t) ∪ P_known(t − 1)
   End Loop
Update k Competitive Templates (Using Best Value Known in Each Objective)
End Loop
```

Figure 4.5 MOMGA-II Pseudo-code [133]

Another reason for choosing the island model is the fact that it allows for migration from one island to another. This intermingling can seed a population that is doing poorly with good values. This results in populations that are equal to, or better than, the serial results, and completed in less time.

4.6 Summary

This chapter describes how the MOMGA-II algorithm works. But to get a better understanding of how MOMGA-II came to be, a discussion of the preceding algorithms must occur. So the chapter opens with a discussion of the MOMGA algorithms. The mGa algorithm, discussed in Appendix C, is the single-objective counterpart to the multi-objective MOMGA. The biggest drawback of these algorithms is the PEI portion. While it was effective, it was not efficient and limited the usefulness of the algorithms. The chapter closed with the newest innovations, the fmGA and the MOMGA-II. These algorithms

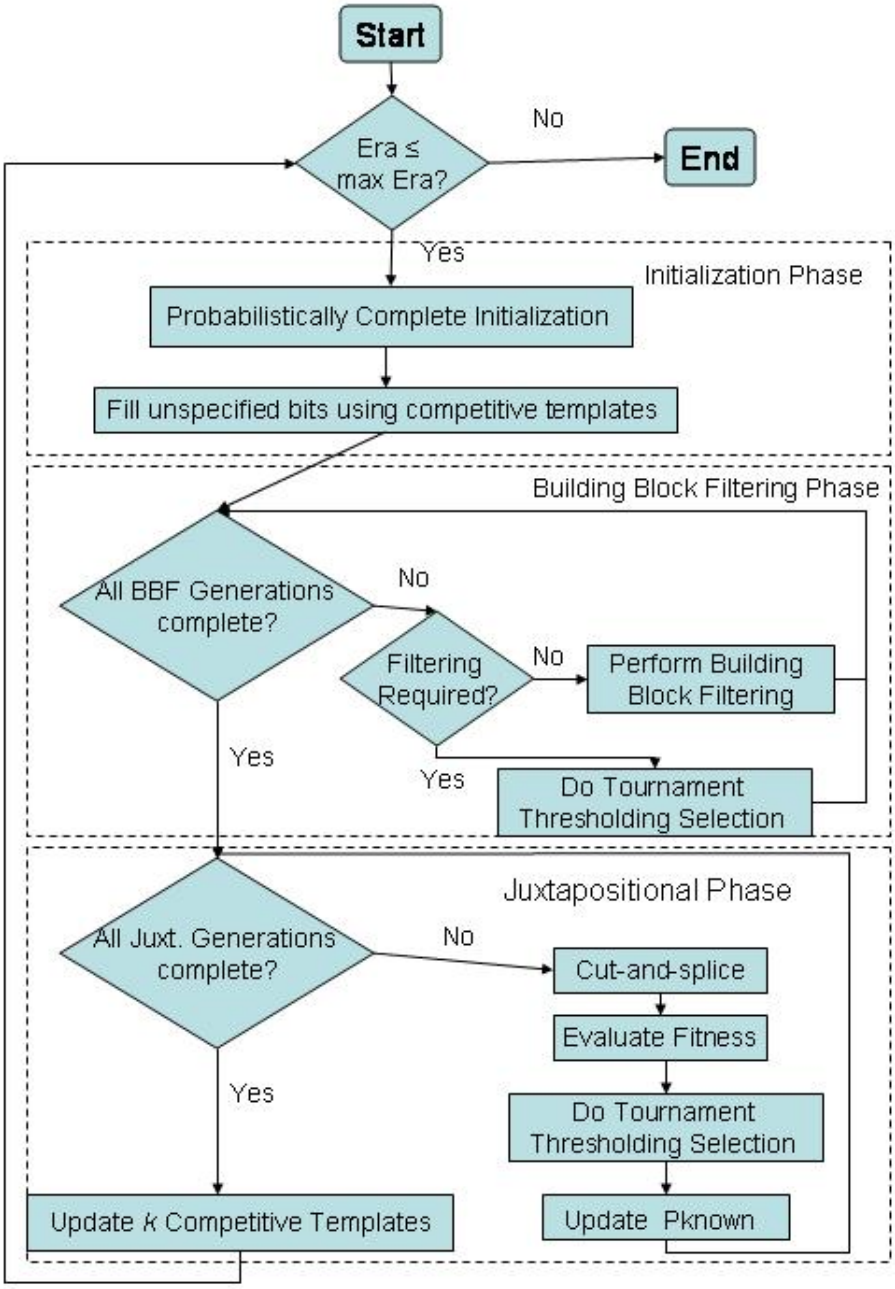

Figure 4.6 MOMGA-II Flow Diagram

address the bottleneck issue discovered in the two previous algorithms and create programs usable on larger problems.

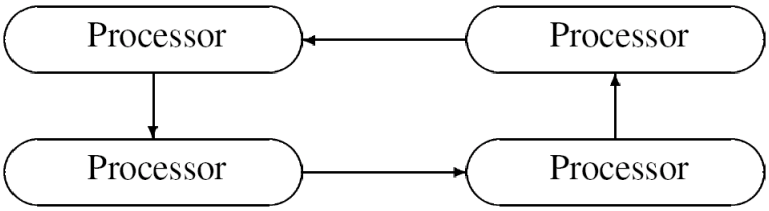

Figure 4.7 Island Model

The next chapter discusses the design of the algorithm. This includes a discussion of how the problem and algorithm domains map together.

5. Software Design

5.1 Introduction

The previous chapters introduced the problem and algorithm domains. This chapter maps the problem domain to the algorithm domain. Section 5.2 discusses the various design options and explains why the options were chosen. Section 5.3 lists some of the assumptions that are made and discusses why these assumptions were made. Section 5.4 states what program was selected and maps the problem to the program. Section 5.5 discusses the implementation language of MOMGA-II and the advantages of using it over the other options. Section 5.6 lists the MOMGA-II parameters used in the experiments as well as the process used to get data into its final format.

5.2 Mapping the problem domain to the algorithm domain

This section lists the design approach chosen as well as some of the other design options that were rejected.

5.2.1 UAV communication mapping. The first objective is mapping the heterogenous UAV communication problem to a more generalized algorithm. Recall that the problem consists of a formation of heterogeneous UAVs with with multiple communication channels with a fixed communication rate on each channel to each of the other UAVs. Why does the problem need to be mapped to a more generalized algorithm? If the researcher blindly implements his algorithm without noticing it matches a more generalized algorithm, he misses out on a pool of knowledge that can enable him to avoid the pitfalls of previous researchers.

There are three general problems that have similarities to the problem at hand.

- Linear Assignment Problem
- Quadratic Assignment Problem
- Multiobjective Quadratic Assignment Problem

5.2.1.1 Linear Assignment Problem. An assignment problem's task is to assign n items (UAVs) to n machines (locations). Symbolically, the linear assignment problem (LAP) can be described as follows [12]:

S_n = Set of all possible assignments (permutations)

$$\begin{aligned} \sum_{i=1}^{n} x_{ij} = 1 \quad \forall \quad j = 1, \ldots, n \\ \sum_{j=1}^{n} x_{ij} = 1 \quad \forall \quad i = 1, \ldots, n \\ x_{ij} \in \{0, 1\} \quad \forall \quad i, j = 1, \ldots, n \end{aligned} \tag{5.1}$$

To map the UAV communication problem to the LAP, the LAP needs to be set as a minimization problem. In addition, a matrix of values need specifying in order to define the distance from one location to another. But this problem is limited because there is no way to include the communication rates of each UAV. This problem can give you a good approximation only if you have communication rates that are equal for each UAV. Since this is not the case for our problem, the LAP doesn't adequately map to the problem.

5.2.1.2 Quadratic Assignment Problem. The quadratic assignment problem is an extension of the LAP, in that it has distances applied to the locations and flows assigned each facility. Recall equation 2.1 which gives a mathematical description of the QAP.

This problem maps well to the UAV problem. It takes into account both the distance between the UAV locations in the formation and the bit rate associated with each UAV. This problem would be perfect with a single communication channel. But since we are using multiple communication channels, each flow is generated independently of the others. Then, a comparison of the results is done in order to find a good solution. While this approach would work, it's not the best approach to take.

5.2.1.3 Multiobjective Quadratic Assignment Problem. Mathematical, the mQAP is defined in equations 2.2 and 2.3.

This problem maps well with the UAV problem. It takes into account the distances between each UAV and the multiple flows. Since it is a multiobjective problem, the mQAP allows the user to pick the result deemed best along the Pareto front. The mQAP was picked because it most closely matched the UAV communication problem.

5.2.2 Simple Problem vs. Real-World Problem. The real-world problem is more complicated. Table 5.1 lists some of the differences between the simple problem and the real world problem. The list is not all inclusive.

Table 5.1 **Comparison of Simple Problem and Real World**

Simple Problem	Real World
Constant distance	Distances apart fluctuate
Constant communication flow	Communication flow varies
Snapshot in time	Formation constantly changing wrt time
All communication flows without error	Have communication problems at times
Any UAV can fly in any location	Some locations are too restrictive for certain UAVs

In reality, the distance between aircraft in a formation fluctuates. But the formation usually remains the same. Because the minor fluctuations in distance have no effect on the problem, a constant, representative distance is used.

The average communication rate between UAVs is deemed more appropriate to model than communication rates that vary depending on the location of the UAVs and their mission. The decision maker decides what communication rate is the most appropriate for the problem. For example, it may be best to use the overall communication rate for one mission and then use the communication rate of a critical segment of a mission during another. This gives the decision maker more options.

This is an a priori process. The formations are determined based on the formation picked for the mission. This problem only figures the best location for each UAV based on one formation. To determine the best locations in multiple formations, the algorithm must be run consecutive times with each new distance matrix as an input.

This problem assumes that the average communication rates are correct and that no error occurs in communication, or the decision maker has accounted for the error rate in the communication rates. This is done keep the algorithm as simple as possible. By including communication specific processes, the algorithm becomes more limited. It is assumed that the user has already used a communication algorithm to come up with the input communication rates.

It is also assumed that any UAV can occupy any location in the formation. In reality this may not be the case. This simplification was made to avoid checking for constraints. This can easily be added later when actual formation data is used with actual constraints.

5.2.3 Input Data. For input data, a researcher usually has two choices: use actual data or data taken from a test suite run by other researchers. The best approach is to first run the program on a test suite and verify the program works well when compared with other programs. If it doesn't, the program can be either tuned for the problem to improve the results, or the program can be scrapped and a different approach attempted. Once the program seems to work well on the test suite, it should be used on actual data. Since there is no actual data for the communication rates for heterogenous UAVs flying in a group, a test suite is the only thing tested against.

Only one standardized test suite is currently available for the mQAP problem [71]. This suite was used as the input data and the results are compared to results obtained from previous researchers. The test suite used for testing is seen in table E.1.

5.2.4 Data representation. The data representation can take on any number of possibilities. Listed below are some of the them.

- Real numbers
- Integers

- Binary numbers
- Permutations
- Parse Trees

The data representation of problem should be determined based on the problem criteria [38]. Knowing this, parse trees and real numbers can be removed from the list. If using integers, permutations would be a good choice. But before making any decision, the researcher should look at the program he plans to use and see if any particular representation works better with it. Since MOMGA-II is based on the fmGA and it is based on a binary representation, it appears that a binary representation is the best fit. (But this doesn't fit too well for the fitness values, so they need to be saved in either integer or real format.) Since the data is all integer data, we conserve space by representing the fitness functions as integers.

Therefore, the representation chosen is a binary representation with integers representing the fitness functions. It might be effective for the binary representation to include encoding of some type, such as Gray encoding or representing a decimal number. The problem with encoding the binary strings is that it creates some infeasible solutions when the permutation is filled and when the competitive template is applied. That being said, the decision was to have 10 binary digits represent each facility. But each bit is randomly placed. In order to determine the permutation, the binary bits are decoded by having the lowest binary number equal facility one and continue assigning values to each facility until all facilities are accounted for. If two facilities have the same representation, the first one in the list gets the lowest number.

5.3 Assumptions and constraints

Several assumptions are made in order to make the problem tractable. This sections discusses some of the assumptions and why they were made. Section 5.3.1 details some of the assumptions made with respect to the UAV formations. Section 5.3.2 discusses the communication assumptions made.

5.3.1 Formation Assumptions. Several assumptions regarding the flight formation of the UAVs were made. Section 5.3.1.1 discusses how the distance matrix is used and some assumptions that were made. Section 5.3.1.2 discusses the assumptions made regarding a static formation.

5.3.1.1 Actual formation distances vs. Test Suite data. For this research, the distance between UAVs are not actual formation distances. The distances used are the distances found in the test suite. While these are not accurate for our problem, they do not adversely affect our results. Once accurate distances are determined, they be used in a distance matrix. The results should be as effective as our previous results.

Another assumption is that one straight line distance is equal to another straight line distance with respect to communication. But in reality, if one UAV is in the communication path of two others, this may adversely affect communications. The distance listed would then give a faulty result. There are two solutions to this, one, the researcher can increase the distance in the matrix to produce a better result. But then the researcher won't know which distance values are actual and which values are adjusted. In addition, he won't know how much the values are adjusted by simply looking at the matrix. The second way is a better way to adjust the distances is by using an additional matrix containing scalar values. These scalar values adjust the distance values so they take into account different types of interference. Then, these values are multiplied to the distance values before they are multiplied with the flow values. This method gives the researcher more control and is easier to adjust on the fly.

5.3.1.2 Static formations vs. dynamic formations. Formation distances are never truly static. The aircraft may fly in formation while approaching their targets, but they may separate, do their missions, and then reform after their missions are complete. This research is interested in minimizing communication transmission times. Since the aircraft are in formation the majority of the time, using these distances is a good approximation of the distances for overall communication. Plus, since every mission has unique aspects pertaining only to it, a general model is best suited to ignore the inconsistencies between various missions.

5.3.2 Communication Assumptions. Many assumptions and simplifications are required in the realm of communications. Section 5.3.2.1 compares the differences in using actual communication rates and test suite data. Section 5.3.2.2 explains why average communication rates are used instead of varying rates. Section 5.3.2.3 discusses how the researcher is able to weigh the importance of various communication channels and can pick the best the solution that best fits the needs of the operation.

5.3.2.1 Actual communication rates vs. test suite data. Using the actual communication rates between the UAVs is ideal. But in order to get an idea of the communication rates, requires the researcher have access to sample communication data based on the situations to be encountered in the mission. Since it is impossible to know this data at this time, using test suite data is better than estimates. Primarily, because test suite data can be compared to other researchers' results. Then the algorithm can be validated by comparing it to previous results. Using estimates doesn't afford the luxury of comparing previous results with your own.

5.3.2.2 Average communication rates vs. varying rates. Average communication rates are used in this research for several reasons. First, by having only one flow rate matrix for the entire flight instead of having many over the course of the flight, the problem is simplified. This simplification lowers the computation time of the algorithm. Plus the information gained by having multiple matrices per flow does not warrant the additional computational time and complexity.

Secondly, the communication rate can be adjusted so that it best fits the most crucial point or points in the flight. The communication effectiveness is most critical at this juncture.

5.3.2.3 Weighting of various communication flows. Since there is more than one communication channel, there is the possibility that one channel has a higher priority than the others. Choosing a solution that takes into account this weighting is important. By using an MOEA to solve this problem, the researcher is able to look at a

range of solutions that are found along the Pareto front. He can then pick a solution that takes into account the priorities of the various channels.

5.4 Program selection

In order to run the mQAP, a multiobjective program either need be created or modified. The decision was made to modify an existing algorithm.

There are several multiobjective algorithms capable of solving this problem. But it was decided to pick an algorithm developed here at AFIT. This decision was an easy one because any problems can be resolved fairly easily because many of the authors and their advisors are available to answer questions regarding the program.

The next decision was to determine which algorithm to use. The MOMGA-II was chosen over other viable option such as GENMOP. MOMGA-II is an explicit building-block GA. This means that it attempts to link building blocks together in an effort to find the best solution. GENMOP uses implicit building blocks, but the linkage of good building-blocks is dependent on the two alleles positions on the chromosome. Crossover can prevent these linkages from propagating effectively toward a solution, especially if elitism isn't used. The MOMGA-II, while being a more complicated algorithm, keeps good building blocks because once they are linked, they cannot be separated via crossover. In addition, MOMGA-II has been tested on classical problems and the 0/1 knapsack problem, an np-complete problem.

Appendix D.1 maps the mQAP symbolically to the MOMGA-II algorithm.

5.5 Implementation Language

The implementation language for MOMGA-II is the *C* language. There are two reasons for choosing this language. First, this program uses the concepts behind fmGA and the MOMGA program. Since the MOMGA program was designed in *C*, extending its capabilities is easier than starting from scratch. MOMGA was built as a multiobjective version of the mGA [117]. The serial version of this code was written in *C* as well [29].

Another reason for programming in *C* is the use of MPI to create parallel applications. MPI gives the researcher a lot of freedom when it comes to designing a parallel architecture. MOMGA-II used MPI to create three different types of parallel environments:

- Master - Slave Model
- Island Model
- Diffusion Model

The added flexibility of MPI allows the researcher to tweak the MPI parameters in order to get a more robust parallel application.

5.6 MOMGA-II Properties

A big part of the success and failure of the MOMGA-II program are the input parameters selected. Section 5.6.1 lists the parameters that provided the most success.

The data coming from MOMGA-II needs further processing in order to extract the meaningful data. Section 5.6.2 describes the process used to get the data into its final form for analysis.

5.6.1 Input parameters. The MOMGA-II code is very complicated. It consists of 13 *.c* files and 5 files that it pulls parameter values from. Tracking the flow of the program can be confusing, and initially finding the parameter values difficult. Table 5.2 lists the main parameter settings and where they are located. The parameters that vary are ones that are either changed based on the problem size or depend on the method used to run the program.

5.6.2 Process used for getting data results. The MOMGA-II program takes the input matrices and outputs the results of its search into four files. Unfortunately, the data is still raw and needs filtering to extract the meaningful data. Figure 5.1 shows the dataflow diagram of the required process to get the final output data.

First, the MOMGA-II program is run and four output files generated. Then the unique and the dat files are converted from binary to integers. Then one of those (depend-

Table 5.2 **MOMGA-II parameter settings**

Parameter	Parameter setting	Found in
MGA_MODE	0 (fmGA)	mga.def
SCALED_FITNESS	0 (off)	mga.def
ELITISM	1 (on)	mga.def
ELITISM_MODE	1 (PFcurrent to next era)	mga.def
FILE_MODE	1 (Rec each gen)	mga.def
Maximization	0 (Minimization)	parameters
Problem_size	Varies	parameters
Number of bits per facility	10	subfunc
Start era	Varies	parameters
Maximum era	Varies	parameters
Elitism percentage	0.25	parameters
Prob of cut	0.02	parameters
Prob of splice	1.0	parameters
Prob allelic mutation	0.0	parameters
Prob of genic mutation	0.0	parameters
Thresholding	0 (No)	parameters
Tiebreaking	0 (No)	parameters
n_a	Varies (For PCI)	parameters
NUM_FUNCS	Varies	mga.h
NUM_TEMPLATES	Varies	mga.h
NUM_DVS	Varies	mga.h
dvs_len	10 (bits per item)	mga.h
Total generations	100	era
Juxtapositional popsize	300	era

ing on the circumstance) is input into the pareto_enum and the Pareto front is found. This occurs for 30 different instances and then these are used to determine how well MOMGA-II did in finding the Pareto front points.

5.7 Data structures used

The data structures used in the MOMGA-II program are very straight forward. This section details the various data structures used.

5.7.1 Representation. Each allele in the chromosome consists of tuples. Figure 5.2 shows an example of how the chromosome is designed. As you can see, each allele has two values associated with it. The first value is the binary value and the second value is

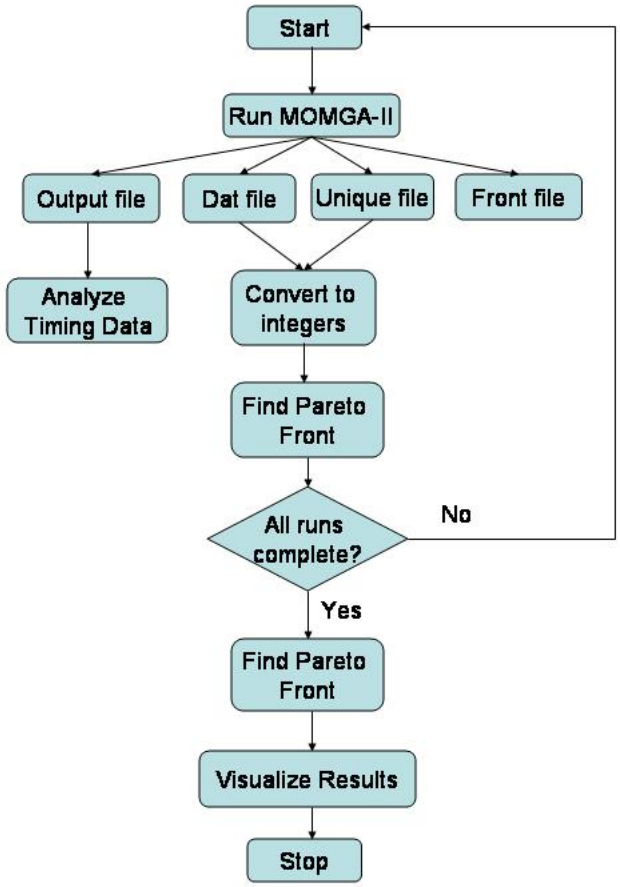

Figure 5.1 MOMGA-II Data Flow Diagram

the chromosome location for the chromosome. This representation allows good building blocks to be grouped together and avoid being divided by the cut and splice method. This is why the fmGA and MOMGA-II are called explicit building block methods. The building blocks are explicitly grouped together along the chromosome.

When these values are evaluated, they are placed on a chromosome template, so that each value is in its proper location. If the chromosome produces a good output, it may take the place of one of the competitive templates. The competitive template fills any empty spaces for an underspecified chromosome. It is hoped that the template helps to guide the search toward better solutions.

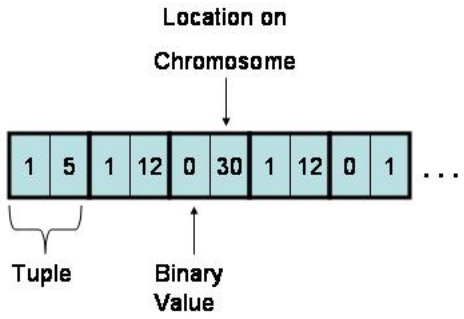

Figure 5.2 Example of a chromosome in the MOMGA-II. Each allele is represented by a tuple.

5.7.2 Input Data. The input data is read in as a matrix. The matrix data is then put into a scalar array. For example, table 5.3 shows a 10×10 input matrix. The columns and rows are labelled 0 though 9. The row value is multiplied by a scalar representing the total number of columns per row. For our example, the scalar value for the row is 10. So if you want the first row and first column, you get the following results:

$$\begin{aligned} r = 0, c = 0 \\ X[r * 10 + c] = X[0 * 10 + 0] \\ X[0] = 0 \end{aligned}$$

Now if you want the value from row 7, column 5, you get the following:

Table 5.3 **Example of an input matrix**

0	10	8	15	11	18	12	4	5	17
10	0	28	7	16	22	14	9	23	9
8	28	0	11	22	15	9	17	15	11
15	7	11	0	16	11	10	18	19	14
11	16	22	16	0	19	22	14	12	27
18	22	15	11	19	0	18	9	7	22
12	14	9	10	22	18	0	11	18	7
4	9	17	18	14	9	11	0	10	17
5	23	15	19	12	7	18	10	0	18
17	9	11	14	27	22	7	17	18	0

$$r = 7, c = 5$$

$$X[r * 10 + c] = X[7 * 10 + 7]$$

$$X[75] = 9$$

5.7.3 Output Data. The output data structure is fairly straight forward. Each row of the output file contains the integer representation of one chromosome. The row also contains the objective function values calculated for that permutation.

5.8 Constraint Handling

The data representation was set-up in such a way that no constraint handling is necessary. Since 10 binary digits represent each UAV, that gives a 1024 possible input values. This makes the chance of having the same binary representations over 10 bits in the chromosome a rare occurrence. Plus, if two similar sets are found, the first set in the chromosome gets the lower number.

In addition, when applying the competitive template to the under-specified chromosomes, no constraint handling is needed. By avoiding the need to repair individuals, the algorithm runs more efficiently since it doesn't have to always check for error conditions that need repair.

5.9 Additional Programs

The MOMGA-II program creates the three data files discussed in section 5.7.3. But most of the results are hard to decipher. Several programs are used to clean up the data and to display it in a meaningful way. The next few sections briefly describe what each program did to the MOMGA-II data and why they were useful.

5.9.1 Main_Short.cc. This program is an exhaustive search program that finds all possible solutions to the mQAP. The program starts by placing UAV one in formation location one, UAV two in formation location two, etc., until all the locations are filled with UAVs. The algorithm calculates the values for all the objectives, and then writes the results to disk. These objective values are then stored as a baseline for other objective values to compare their results. The algorithm swaps the UAVs in the last two locations and then recalculates the objective values. These results are compared to the stored results. If all objective values are worst than the stored results, they are not saved and another permutation is calculated. If some of the results are better than the stored result, the results are written to disk. If all results are better than the saved result, the data is saved to disk and the results are saved for all future comparisons. This continues until all permutations have been tried. Once the data file reaches a certain size, the main_pareto program is run to pull out the current Pareto Front. At the end of the algorithm, the final Pareto Front is derived using the main_pareto algorithm.

5.9.2 qapbin2int.c. This program takes the string of binary numbers that represented the UAV numbers and replaced them with their appropriate integer value. For example a 10 location and 10 facility problem, with 2 flows, would have 102 values. The first 100 values are the binary numbers. Every 10 digits represents one UAV. The last two values are the total flow results. As the program runs, it calculates the integer value for the first ten binary digits. It then sets those equal to the lowest number found. The next ten digits are grabbed and compared to the lowest found. If the new value is lower, it replaces the old value. At the end, the lowest number is found and a one is placed into a sorted matrix at the location it was found in the input matrix. This value is then set to a high value that is beyond the scope of the other numbers. The process is then repeated until all

of the binary numbers are sorted and the new matrix displays the results in integer form. Then the two flow totals are added to the end of the row to show the values obtained using the permutation. If two UAVs have the same binary values for all 10 digits, then the first one in the list is given the lower number.

5.9.3 SeparateBBSize.c. This program divides a run that uses more than one building block size into separate files. Each file only contains the results of one particular building block size. This program is only used when comparing runs with only one building block size to runs with many building block sizes.

5.9.4 main_pareto.c. This program cleans up the Pareto front generated via MOMGA-II. MOMGA-II generates many duplicate results for its Pareto front analysis because it only looks at the binary representation and not the integer representation. Since more than one binary string can equal the same integer string, there are duplicates generated. This program runs on the integer representation so all of the extemporaneous data is removed.

In addition, this program combines results from the 30 individual runs and combines the results of separate building block sizes.

5.9.5 MATLAB. MATLAB is used to graphically display the results of the data runs. This program is very useful to aid the researcher in determining how one run compares with another. In addition, it provides a great scientific visualization tool to aid the researcher in determining if his results are on target. In this research, MATLAB aided in the discovery of how the results propagated toward the true Pareto Front. This visualization showed flaws in the original algorithm and gave clues about how to fix the algorithm to overcome the problems.

5.10 Summary

Previous chapters discussed the program used in this research as well as the mQAP. This chapter's main goal is to link the two.

This chapter discusses the design decisions taken for this research. The discussion focusses on the options that have been chosen and the reasons why certain options are discarded and others are implemented. Presented next is a description of the implementation language, along with the reasons for its selection. Described next are the parameter values used for MOMGA-II. These values give future researchers the ability to duplicate these findings. The chapter also discusses the data representations associated with the problem. Also mentioned are the other programs used to filter out the meaningful data from the runs.

The next chapter describes the design of experiments used in this research.

6. Design of Experiments

6.1 Introduction

The main goal of this research is the creation of an effective and efficient algorithm that renders good solutions for minimizing the total propagation time delay for a formation of UAVs. But to achieve that goal, several smaller goals need addressed. These goals can be classified under two categories: effectiveness goals and efficiency goals. Section 6.2 lists the the effectiveness goals and the experimental design used to validate that these goals are met. Section 6.2.3 lists the experimental design used to validate the efficiency goals. Finally, section 6.4 states the computing environment that the experiments were conducted.

6.2 Effectiveness Goals

Finding good solutions is the top priority for this research. Therefore we have to validate that our algorithm does indeed find good solutions. Since the mQAP is a relatively new construct, we only have limited data to compare our algorithm. But since the 10 location and 10 UAV test problems can be solved to optimality, we use this as a benchmark to validate that our algorithm is effectively finding solutions along the true Pareto Front.

Once a baseline set of runs are completed and analyzed, algorithm parameters can be tweaked to improve effectiveness. This section lays out how the experiments were conducted in order to improve the effectiveness of the algorithm.

6.2.1 Find optimal results. Before we run MOMGA-II, we need something to compare it with. Knowles and Corne state how many points they found on the True Pareto Front using an optimal method [71, 72]. While this information is useful, it doesn't state enough information for our purposes. Therefore, an optimal algorithm was created and run on the ten UAVs and ten location problems. This algorithm goes through every permutation of UAVs and locations and finds the values for all the communication flows. When completed, the results are input into a program that pulls out the Pareto Front. These results are then used to compare to the MOMGA-II results.

For the 20 and 30 location problems, a comparison must be done to Knowles and Corne's results. In addition, we apply some of the MOEA metrics mentioned in Section 3.3.2.

6.2.2 Initial MOMGA-II runs. The MOMGA-II was previously run on the multiobjective knapsack problem. For it to work with our problem it has to be modified to handle the mQAP. This required replacing the knapsack portion of the code with code that calculates the values of the mQAP. Once the modifications are complete, the program is run to get some baseline results. These results are then compared to the optimal values found in the 10 location problems. In the 20 and 30 location problems, we compare the results to the ones found by Knowles and Corne. If the results prove to be effective, then we can avoid tweaking parameters. If the results are not effective, some parameter changes need to be made in order to improve the effectiveness of the algorithm.

6.2.3 Validate the effects of the competitive templates and building-block sizes. In addition to validating that this is an effective algorithm for our problem domain, we also want to gain further insight into how the algorithm works. Since this is an explicit building-block GA it is helpful to see if certain building-block sizes gravitate to certain locations on the Pareto Front.

In order to do this and ensure that the competitive templates do not play a role in the larger building block sizes, we need to redesign the way the algorithm works. Some modifications were made to the code in order to enable it to run only one building-block size at a time. We couldn't just separate out the building-block sizes from the previous runs in order to get these results for two reasons. First, since elitism was used, population members were passed from one generation to the next. So at the very least we would have to rerun the program with elitism off in order to avoid the propagation of individuals. But even if we did that we still have a second problem. The competitive templates not randomized between building block sizes. That means competitive templates from previous building block sizes would be passed on to larger building-block sizes. Because of these reasons, the code had to be rerun in a manner so that only one building-block size is run at a time and the competitive template is randomized before each new building block size.

Now that we have runs that have a competitive template that is passed on from smaller building block sizes to larger ones, we can compare those results to our new results to see if one method is better than the others.

Efficiency goals: While the effectiveness of our algorithm is our primary concern, we don't want to turn our back on efficiency. If our algorithm is very effective, but it takes too long to run, then we may be better off finding another algorithm that can get us our results in an expedient manner.

Initial runs: The timing data of the initial runs is recorded. This data is our baseline data. In addition, the optimal program's time is recorded. Since the optimal program cycles through all possible results, it's efficiency is poor. But it has been improved in order to only record "good" results. This increases the efficiency for the Pareto Front analysis. These two sets of time are used as our baseline time. Our goal is to attempt improve the efficiency of our algorithm while we maintain or improve the effectiveness.

Look for time saving approaches Once the baseline time is derived, we need to delve into the program to see if there are any bottlenecks. One bottleneck appears to occur due to all the writing to the data file. By limiting the writes, we may be able to increase our efficiency.

6.2.4 Parallel approach. The parallel approach should save time just because it is running multiple instances at the same time. We want to get the timing data of these runs and compare them to timing data of the serial runs. The efficiency and speedup are charted and show if MOMGA-II is scalable or not.

6.3 Metrics Used

In order to quantify the effectiveness and efficiency of the experiments, metrics need to be established and used. This section discusses the metrics implemented and why they were chosen.

6.3.1 MOEA Metrics Used. Section 3.3.2 lists many of the MOEA metrics that are currently used. For this research, when comparing to PF_{true}, the error ratio was chosen

as the metric. This metric best captures the effectiveness of the algorithm being tested. Many of the other metrics can be deceiving. Metrics that just contain an aggregate of the points can make one result appear to be superior because it has more non-dominated points, when in reality, many of those points are dominated by the competing result. The generational metrics are nice, but are not necessary.

For comparing two populations with no PF_{true}, the two set coverage metric is used. This metric states the percentage of points dominated in one population by another. This metric best describes the effectiveness goals for this research.

6.3.2 Parallel Metrics Used. Appendix B lists some of the parallel metrics used in the research field. Speedup and efficiency are the two metrics chosen for this research. These metrics best capture the efficiency goals for parallel processing. Speedup can quantify how much faster parallel processing completes a task when compared to serial processing. Efficiency can quantify how well the processors are being utilized. All of the other metrics state basically the same thing, but in different ways.

6.3.3 Building Block Metrics Used. For the building block portion of the experiments, there are currently no metrics used in the literature. So, in order to quantify how building block size affects the outer edges of the Pareto Front, a new metric is created.

For this metric, the range of each objective function value is halved. Then, each Pareto Front member located in the upper segment of each objective function is counted. Then the ratio of the total number of large building block members to the total number of small building block members is calculated. This result is a number. If the results is greater than one, there are more large building blocks on the outer edge of the Pareto Front. If the number is between zero and one, then there are more smaller building blocks located on the outer edge. Equation 6.1 shows the basic equation for this metric.

$$BBLocationMetric = \frac{\sum BBSize_{large} \in PF_{outer}}{\sum BBSize_{small} \in PF_{outer}} \tag{6.1}$$

where the outer edge of the Pareto Front is defined as the upper half of the possible values for the objective function. For example, if an objective function ranges between values 2000

and 6000, all population members located on the Pareto Front between 4000 and 6000 for that objective function are counted. Note that for other types of Pareto Fronts, the lower half of the objective function may be more appropriate, depending on the way the Pareto Front is shaped. For this research building block sizes of one and two are considered small. Building block sizes of nine and ten are considered large.

6.4 Computing Environment

The computing environment can be broken down into two segments the hardware environment and the software environment. Section 6.4.1 lays out the hardware used. Section 6.4.2 lists the important software used.

6.4.1 Hardware Properties. The hardware systems used for the testing can be seen in table 6.1. While the backplanes are similar for both systems, the Polywells use faster AMD processors than the ASPEN machines. They also have larger cache memories. A disadvantage is their memory size is only one-fourth the size of the ASPEN machines. In addition, the ASPEN system has more nodes, and they also have two processors per node vs. one processor per node for the Polywells.

Table 6.1 **System Hardware Configurations**

Variable	Cluster 1 (ASPEN)	Cluster 2 (Polywells)
Processors	Dual PentiumIII, 1GHz	AMD Athlon, 1.4GHz
Cache(L1 I,D/L2)	(16,16/256)KB	(64,64/256)KB
Backplane	Fast Ethernet	Fast Ethernet
RAM	1 GByte	768 MByte
Switching	Crossbar Switch	Crossbar Switch
Diameter	1	1
Bisection Width	p	p
Arc Connectivity	1	1
Cost	p^2	p^2
Disk I/O	RAID 5	RAID 5
Memory type	Distributed	Localized
Node Specifics	48 nodes 2 CPUS/node	16 nodes 1 CPU/node

In order to keep our timing consistent, all of our timed runs occur on the ASPEN machines. We do this because ASPEN utilizes a scheduling system that reserves the processors for you when you submit a job. This prevents other processes interfering with your results. The Polywells don't incorporate a scheduler, so at anytime during your run, another process may step on yours and result in erroneous timing data.

6.4.2 Software Properties. There is little difference between the two systems as far as software is concerned, as Table 6.2 shows. The one difference in the operating systems. The Polywells use Redhat version 7.1 while the ASPEN machines use a newer version of Redhat, version 7.3. In all of the runs we have done with other software projects, we have not noticed an instance where a program runs on one system but not on the other. All the rest of the software is identical on both systems.

Table 6.2 **System Software Configurations**

Variable	Cluster 1 (ASPEN)	Cluster 2 (Polywells)
Operating System	Redhat Linux 7.3	Redhat Linux 7.1
MPI Compiler	mpich-1.2.1	mpich-1.2.1
gcc Compiler	Version 2.96	Version 2.96
g++ Compiler	Version 2.96	Version 2.96

6.5 Summary

This chapter lays out the experimental design employed for this research. Specifically discussed are the effectiveness and efficiency goals and the steps taken to achieve them. A discussion of computing environment, both from the hardware and software side is presented. The next chapter lists the results of the experiments and presents an analysis of these results.

7. Results and Analysis

7.1 Introduction

This chapter presents the results and the analysis of the data. Section 7.2 lists the results of the first experiments run. These experiments are the baseline. Section 7.3 shows the results of the parallel experiments. Section 7.4 contains the results and analysis of the building block and competitive template experiments.

7.2 Baseline Experiments

This section lists the initial results and analysis. The test suite described in section 7.2.1 is used not only for the initial experiments, but for all the experiments.

7.2.1 Test Suite. Since the mQAP is a newly formulated problem, only one standardized test suite is currently available, as previously stated [71]. This data was used as the input data and the results were compared with actual results or the results obtained from previous researchers. The test suite used for testing is seen in Table E.1.

7.2.2 Experiment parameters. The parameters used for these experiments include those in Table 7.1 as well as those described in Section 5.6.1.

Table 7.1 **Population sizes for N facilities and locations**

	Population by (N)			# of Generations by (N)		
Era	(10)	(20)	(30)	(10)	(20)	(30)
1	403	401	400	300	100	300
2	413	405	402	20	100	20
3	430	411	405	20	100	20
4	455	419	408	20	100	20
5	491	431	413	20	100	20
6	553	458	431	20	100	20
7	601	464	427	20	100	20
8	685	487	436	20	100	20
9	794	514	447	20	100	20
10	937	546	458	20	100	20

7.2.3 Results. The MOMGA-II results are taken over 30 data runs. The hardware configuration for the experiments can be found in table 6.1. Table 6.2 lists the software configuration.

Some of the baseline results are shown in figures 7.1 - 7.4. Appendix F lists all the baseline graphs created.

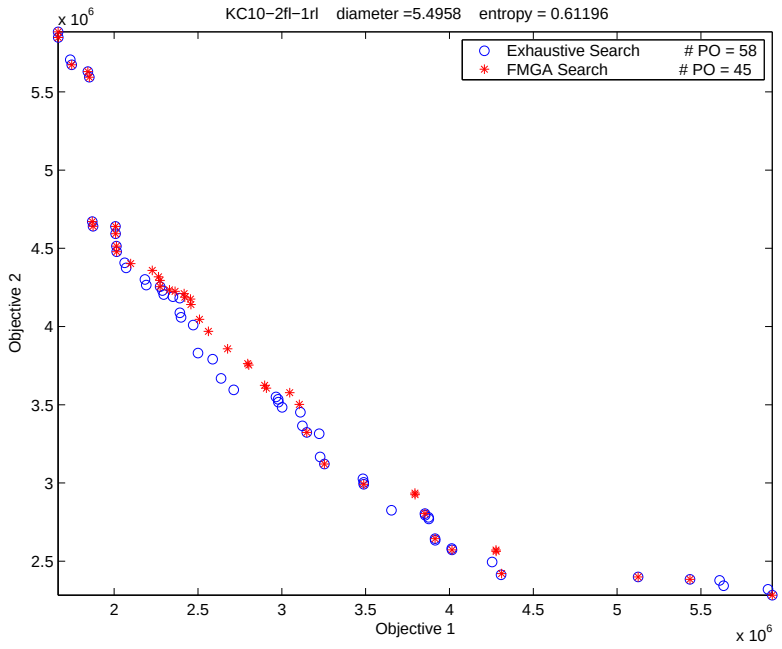

Figure 7.1 Pareto front found for the KC10-2fl-1rl test instance

Table 7.2 shows how the MOMGA-II algorithm performed compared to the data from Knowles [70]. The results are not good. The ten UAV instances are better than the MOMGA-II because those are the PF_{true} values, so those can be expected. But the MOMGA-II missed many points. In addition, Knowles found many more nondominated points than the MOMGA-II found for the harder instances.

Table 7.3 shows the results of MOMGA-II compares with the PF_{true}. The MOMGA-II performs poorly in five of the eight, it could not find more than 45% of the points on PF_{true}. The mean number of points found on the True Pareto Front is only 54%. These results indicate that the MOMGA-II is not performing effectively, so changes must be made either with the parameters or with the algorithm itself.

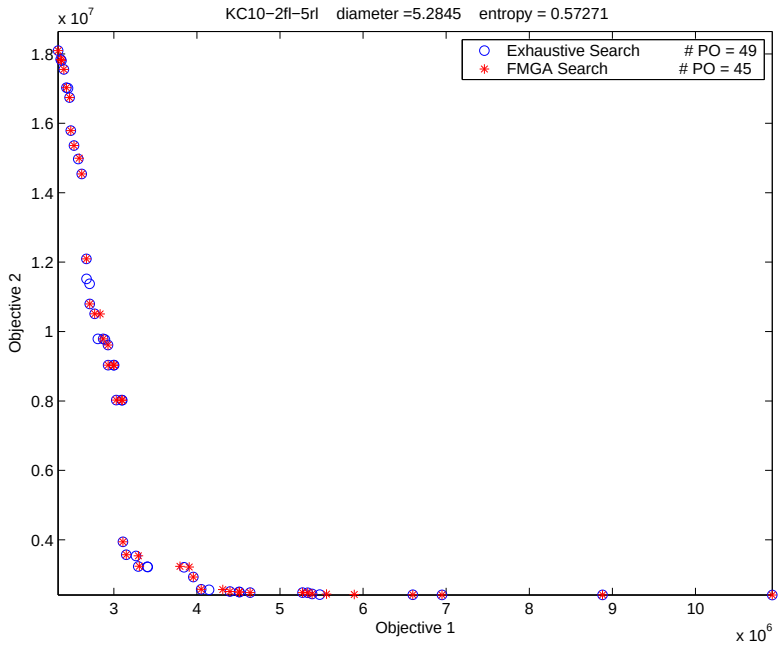

Figure 7.2 Pareto front found for the KC10-2fl-5rl test instance

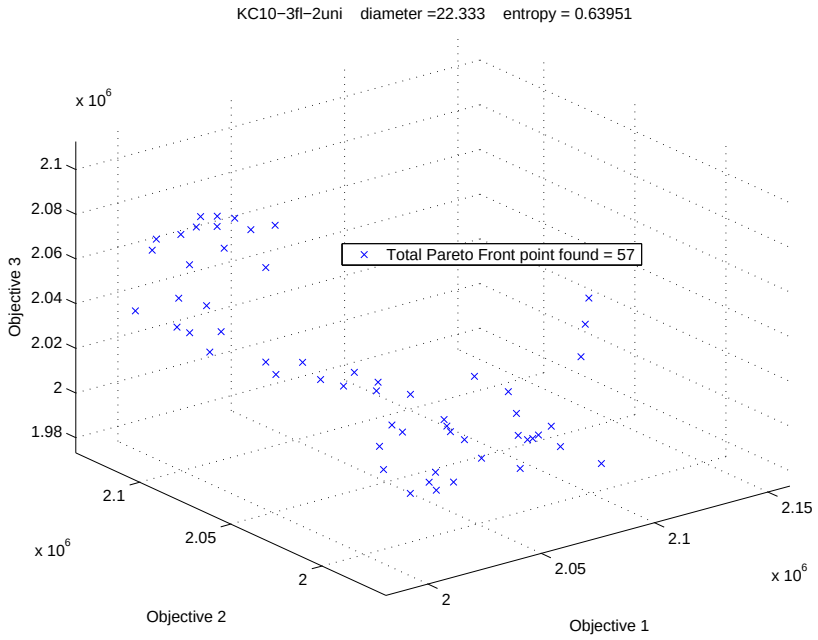

Figure 7.3 Pareto front found for the KC30-3fl-2uni test instance

7.3 Parallel Experiments

These experiments compared the parallel results with the serial results and with the results found using a deterministic algorithm. See Section 5.9.1 for a description

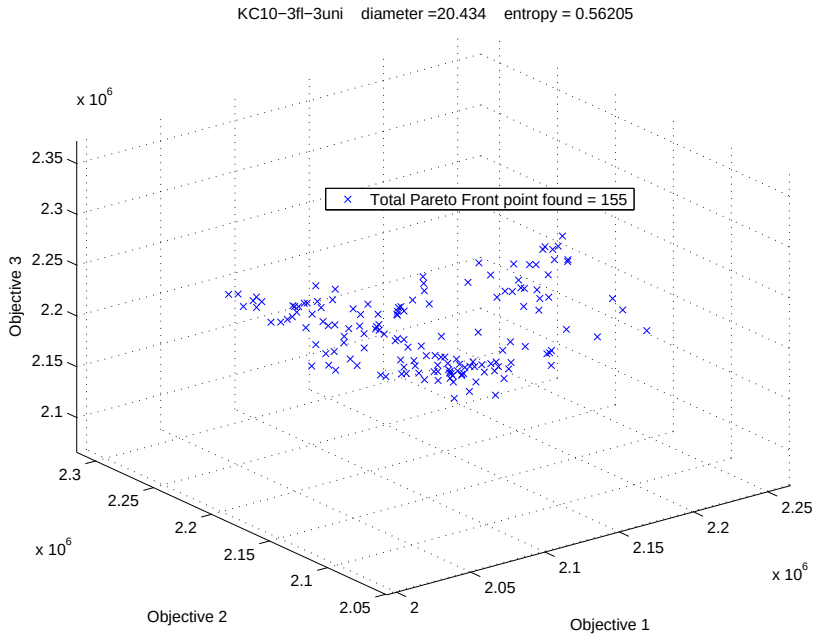

Figure 7.4 Pareto front found for the KC30-3fl-3uni test instance

of the algorithm. The MOMGA-II results are taken over 30 data runs. The hardware configuration for the experiments can be found in table 6.1. Table 6.2 lists the software configuration. The parameters used for the baseline runs are maintained in order to get accurate timing results.

The metrics used for the parallel experiments are speedup and efficiency. Speedup is chosen for its commonality throughout the literature and its ability to show how much faster or slower the parallel processing is compared to serial processing. Efficiency is used to show the amount of time the processing element is used vs. its idle time. Table 7.4 lists the results of the speedup and efficiency analysis. By using more processors, the speedup is increased enabling the runs to be done in a more efficient manner. The efficiency for each processor decreases as more processors are added. This means the processors are in idle mode more often while waiting to do processing. This means there will be scalability problems as more processors are added.

Figure 7.5 shows a graph of the mean time to finish an experiment with set population sizes. This indicates that as more processors are added to search for solutions there is almost a linear speedup when compared to running the same number of searches in serial.

Table 7.2 **Comparison of MOMGA-II Results to Knowles Results**

	Knowles Results			MOMGA-II Results		
Test Name	# ND pts	Diameter	Entropy	# ND pts	Diameter	Entropy
KC10-2fl-1uni	13	7	0.71	13	5	0.69
KC10-2fl-2uni	1	6	0.39	1	0	0
KC10-2fl-3uni	130	8	0.78	118	6	0.87
KC20-2fl-1uni	80	15	0.828	24	11	0.82
KC20-2fl-2uni	19	14	0.43	538	15	1.48
KC20-2fl-3uni	178	16	0.90	51	12	0.92
KC30-3fl-1uni	705	24	0.97	126	20	0.50
KC30-3fl-2uni	168	22	0.92	58	22	0.64
KC30-3fl-3uni	1257	24	0.96	155	20	0.56
KC10-2fl-1rl	58	8	0.68	44	5	0.61
KC10-2fl-2rl	15	7	0.49	10	5	0.56
KC10-2fl-3rl	55	8	0.62	36	6	0.71
KC10-2fl-4rl	53	8	0.58	34	4	0.53
KC10-2fl-5rl	49	8	0.63	45	6	0.69
KC20-2fl-1rl	541	15	0.63	17	12	0.73
KC20-2fl-2rl	842	14	0.6	12	11	0.76
KC20-2fl-3rl	1587	15	0.66	29	12	0.91
KC20-2fl-4rl	1217	15	0.51	25	10	0.18
KC20-2fl-5rl	966	15	0.56			
KC30-3fl-1rl	1329	24	0.83	191	24	0.79
KC30-3fl-2rl	1924	24	0.86	183	24	0.77
KC30-3fl-3rl	1906	24	0.86			

Table 7.3 **Comparison of MOMGA-II Results to PF_{true}**

	True Pareto Front Points		
Test Name	MOMGA-II	Deterministic	% Found
KC10-2fl-1uni	9	13	69
KC10-2fl-2uni	1	1	100
KC10-2fl-3uni	40	130	31
KC10-2fl-1rl	21	58	36
KC10-2fl-2rl	5	15	33
KC10-2fl-3rl	23	55	42
KC10-2fl-4rl	24	53	45
KC10-2fl-5rl	36	49	73
Mean	53.76		
Std. Dev.	24.59		

Table 7.4 **Speedup and Efficiency Results**

Number of Processors	Speedup	Efficiency
2 Processors	1.842	0.9210
4 Processors	2.376	0.5941
8 Processors	3.431	0.4289

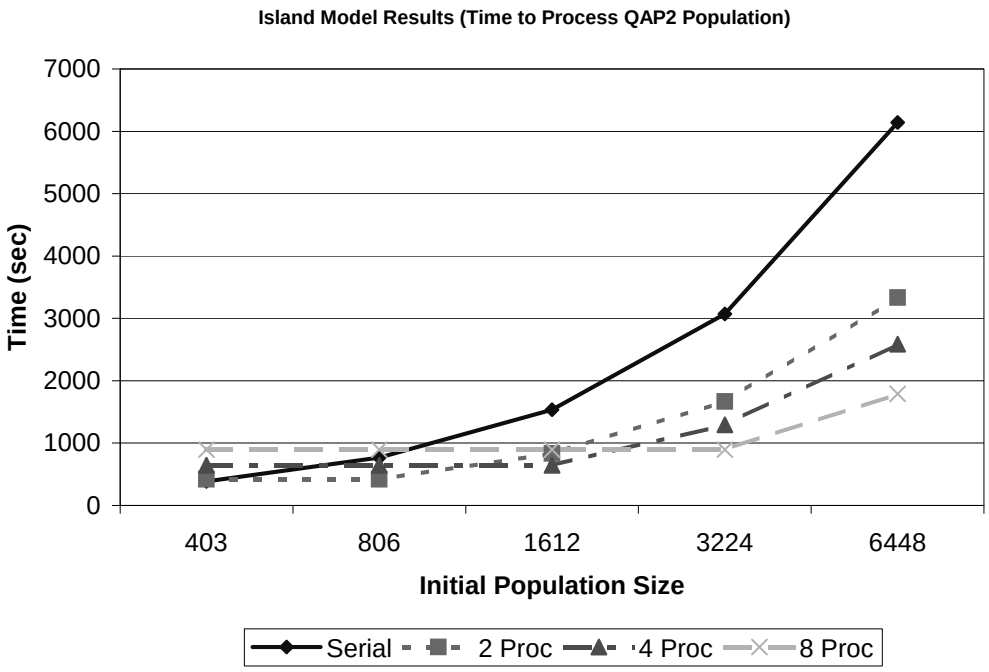

Figure 7.5 Speedup results from running MOMGA-II on data set KC10-2fl-1uni.

7.4 Building Block and Competitive Template Experiments

7.4.1 Effectiveness of New Method. Since the baseline method performs poorly, a new method or parameter set is tried. It is decided to create a new method of running MOMGA-II. This method runs only one building block size at a time. The former method starts at a small building block sizes and finishes at a larger size. With each new building block size, the competitive template is kept. This guides the search toward the best known solutions. With the new method, the competitive template is randomized with each new building block size. This allows for more exploration in the algorithm and helps to avoid premature convergence.

Figures 7.6 - 7.11 show the results of the experiments. These results are also contained in Table 7.5. The results show that the new method performs much better than the old

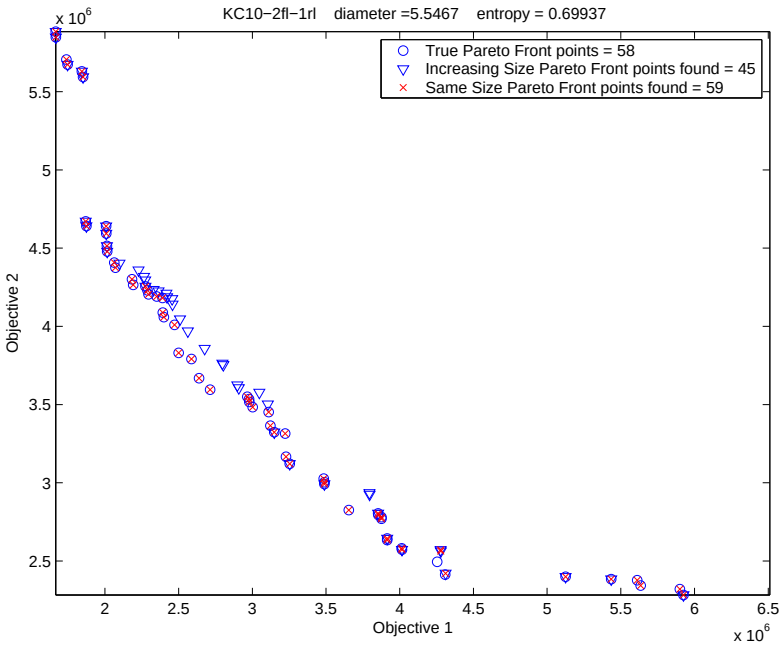

Figure 7.6 Comparison of MOMGA-II methods to optimal results on KC10-2fl-1rl test instance

method on all instances except one. The one time the old method performs better is when there is only one data point as a solution. These results show that, with the exception of one instance, the new method is more effective than the old method. See Section 7.4.3 for further analysis on why the new method performs better than the old method.

7.4.2 Building Block Size Results. In order to analyze the building block sizes and their proximity on the Pareto Front, we graphed each building block size individually. This resulted in longer execution times for the algorithm but shorter overall execution time because the Pareto Front analysis took much less time with smaller population sizes. The Pareto Front analysis algorithm has a complexity of $O(kM^2)$ where k is the number of objectives and M is the population size. So when the population size is divided into 10 smaller chunks, the Pareto Front analysis algorithm runs much faster due to the smaller population sizes. Then we compared the two methods to the True Pareto Front. Table 7.6 shows how many building block sizes found the Pareto Front. It is interesting to note that the randomized template appears to do better for smaller building block sizes while the propagated templates do better on larger building block sizes. One possibility for

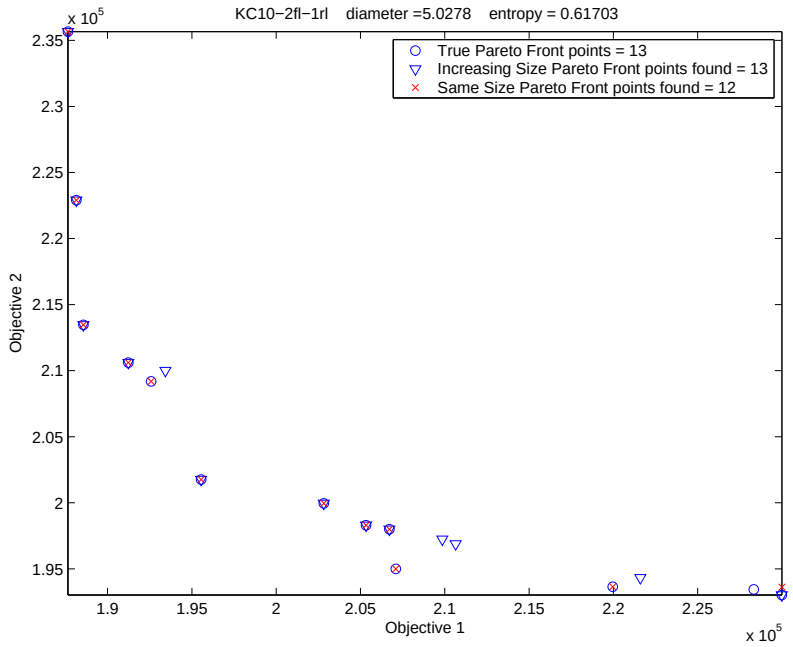

Figure 7.7 Comparison of MOMGA-II methods to optimal results on KC10-2fl-1uni test instance

Table 7.5 **Comparison of MOMGA-II Methods to PF_{true}**

	True Pareto Front Points				
Test Name	Deterministic	Old Method	% Found	New Method	% Found
KC10-2fl-1uni	13	9	69	11	85
KC10-2fl-2uni	1	1	100	0	0
KC10-2fl-3uni	130	40	31	122	94
KC10-2fl-1rl	58	21	36	56	97
KC10-2fl-2rl	15	5	33	11	73
KC10-2fl-3rl	55	23	42	50	91
KC10-2fl-4rl	53	24	45	47	89
KC10-2fl-5rl	49	36	73	49	100
Mean		53.76		78.49	
Std. Dev.		24.59		32.75	
Mean (w/o anamoly)		47.16		89.70	
Std. Dev. (w/o anamoly)		17.28		8.82	

this is that since there are more population members for larger building block sizes, more exploration is done with the old method and it is able to locate more points on the Pareto

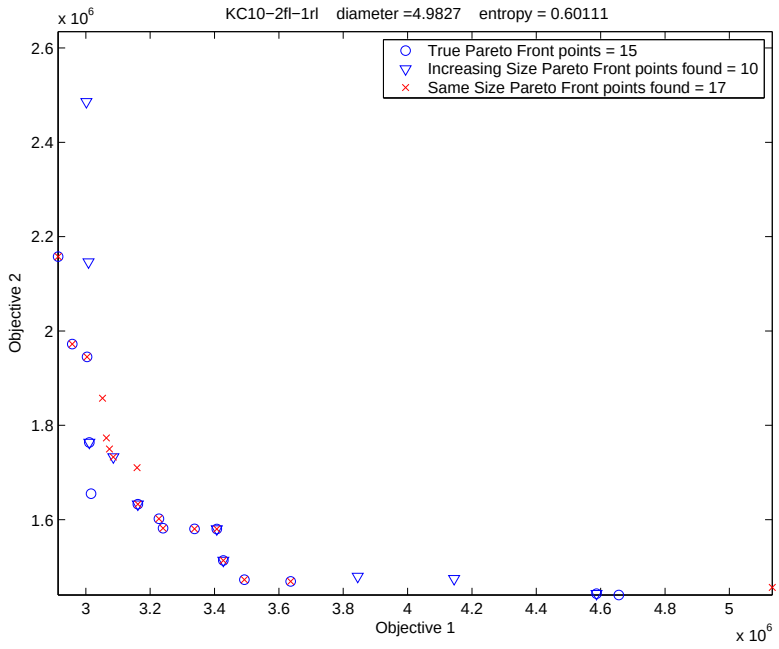

Figure 7.8 Comparison of MOMGA-II methods to optimal results on KC10-2fl-2rl test instance

Front. Plus, the old method has the advantage for finding the outside points of the Pareto Front due to the competitive templates guiding them there.

Table 7.6 **Number of Building Block Sizes Located on the True Pareto Front**

Building Block Size	Pareto True	Randomized Comp. Temp.	Propagated Comp. Temp.
BBSize 1	13	4	4
BBSize 2	13	7	3
BBSize 3	13	6	4
BBSize 4	13	6	4
BBSize 5	13	2	3
BBSize 6	13	5	4
BBSize 7	13	5	4
BBSize 8	13	4	4
BBSize 9	13	3	4
BBSize 10	13	1	4

Table 7.7 shows how four of the test instances compared when the building block location metric is used to gauge the number of building blocks on the outer edges of the Pareto Front. On average, about twice as many large building blocks populate the

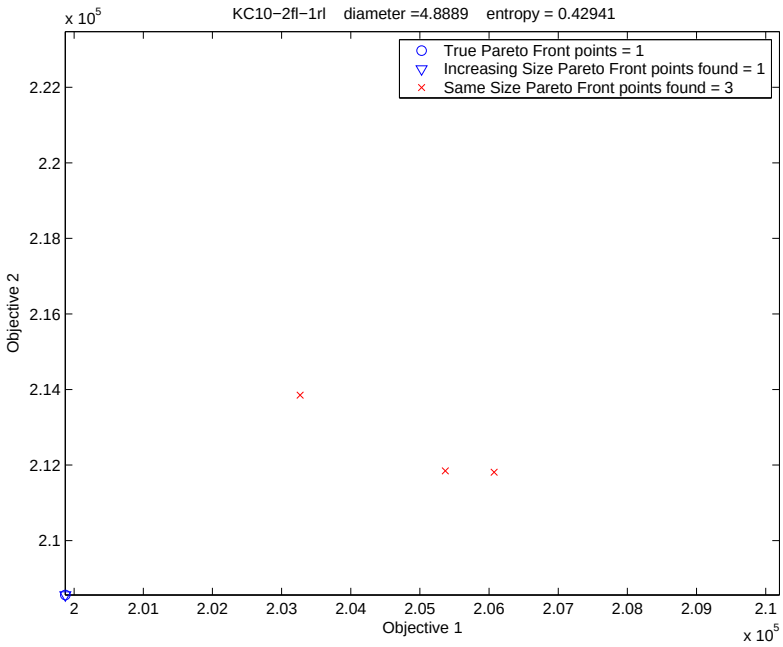

Figure 7.9 Comparison of MOMGA-II methods to optimal results on KC10-2fl-2uni test instance

outside of the Pareto Front than the smaller building block sizes do. This is due to more bits being set in the genotype domain and allows for a better solution in the phenotype domain. Another contributing factor is the larger population sizes that are generated for the bigger building block sizes. These larger populations are generated because of the PCI formula that attempts to set the population to a good statistical sample. Since there are many more permutations of 10 building blocks than 1 building block, there has to be a larger population. These two factors are the primary reason the larger building block sizes are capable of reaching the outer edges of the Pareto Front and the smaller building block sizes are not as capable.

Figures 7.12 through 7.20 show some of the results of the experiments. What is interesting to note is how well the randomized template method found the inner points on the True Pareto Front for building block sizes 4 and 5 but tended to drift from them as the building block size increased. The diameter of the Pareto Front also appears to increase with larger building block sizes. This supports the previous findings of Zydallis [134] and Van Veldhuizen [117].

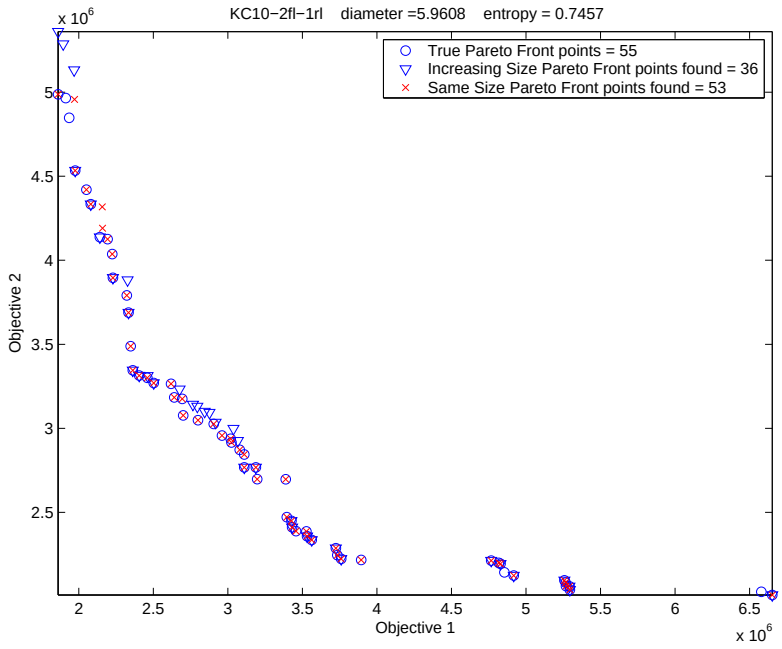

Figure 7.10 Comparison of MOMGA-II methods to optimal results on KC10-2fl-3rl test instance

Table 7.7 **Comparison of Building Block Sizes and Location on Outer Edges of Pareto Front**

Test Name	# Large BB Size	# Small BB Size	BB Location Metric
KC10-2fl-1rl	62	44	1.409
KC10-2fl-1uni	12	3	4.000
KC10-2fl-2rl	4	4	1.000
KC10-2fl-3rl	32	17	1.882
Mean	2.073		
Std. Dev.	1.334		

Two things can be done to help the larger building block sizes reach the True Pareto Front. Adding one or more competitive templates on the interior of the Pareto Front could bring more values toward the center. Another possible method would be to increase the number of generations run for the larger building block sizes. This allows the population more time to converge to the True Pareto Front.

7.4.3 Competitive Template Results. The randomized competitive template method (new method) did better when compared to the propagated competitive template (old method). These results suggest that by randomizing the competitive template,

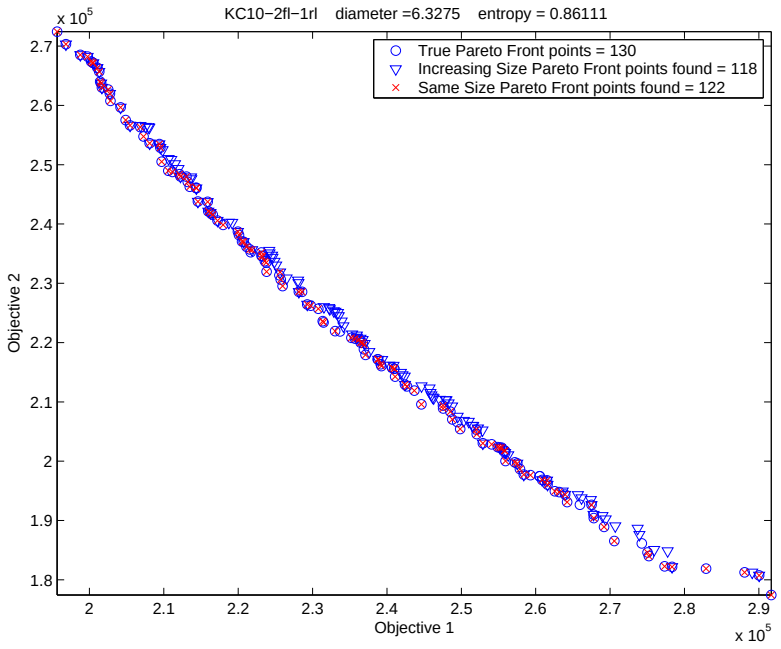

Figure 7.11 Comparison of MOMGA-II methods to optimal results on KC10-2fl-3uni test instance

the algorithm is able to explore the objective space more effectively and yield better results. The better results appear to be due to the placement of the competitive templates and the method for which they are chosen to be applied to underspecified chromosomes. The algorithm randomly chooses a competitive template to apply to the underspecified chromosome. This is similar to the criterion selection technique outlined in section 3.3.1.3. A criterion selection divides its population into subpopulations and then selects the next generation from each subpopulation based on one objective function. These members are placed back into the main population and after the population is mixed and evolutionary operations are applied to the members, the population is subdivided again. MOMGA-II also has competitive templates located at the ends of the Pareto Front, creating a similar circumstance as seen in the criterion selection method. Therefore it is believed that the MOMGA-II suffers from "speciation" much in the same manner as criterion selection methods do. This can be overcome by adding some competitive templates near the center of the Pareto Front.

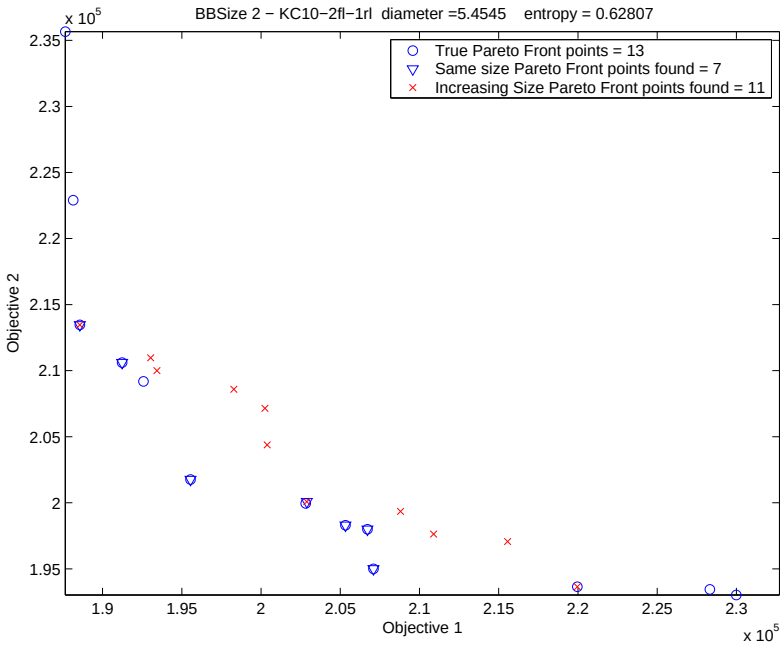

Figure 7.12 Comparison of the effectiveness of using an inherited competitive template BB size 2 vs. initially randomized templates

The results from Table 7.5 support these findings. Whenever there were many points to find, the new method always found more than the old method. The reason why the old method performed better than the new method when there was only one point to find is due to the fact that both competitive templates are pointing at the same location. This directs the search in the same direction as opposed to dividing the search into two directions. Since the new method doesn't have this directed search passed on to the larger building block sizes, they start at a disadvantage when trying to find one or two points.

7.5 Summary

This chapter discusses the results of the experiments. The baseline experiments were found to be not very effective. By adding parallelism to the algorithm, the speedup is increased. This results in our algorithm completing sooner, thus it runs more efficiently. A new method is presented and shows to be far more effective than the old method in most instances. Experiments conducted on building block size and on the competitive template show that building block size does play a small role in finding points along the Pareto

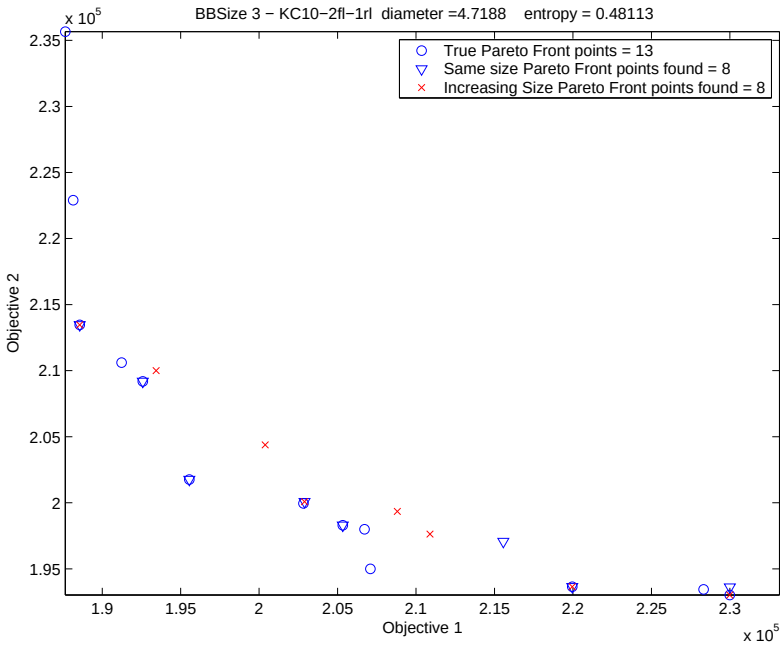

Figure 7.13 Comparison of the effectiveness of using an inherited competitive template BB size 3 vs. initially randomized templates

Front. The competitive template results show that a randomized template at the start of each new building block works the best.

The next chapter discusses the conclusions made from the results.

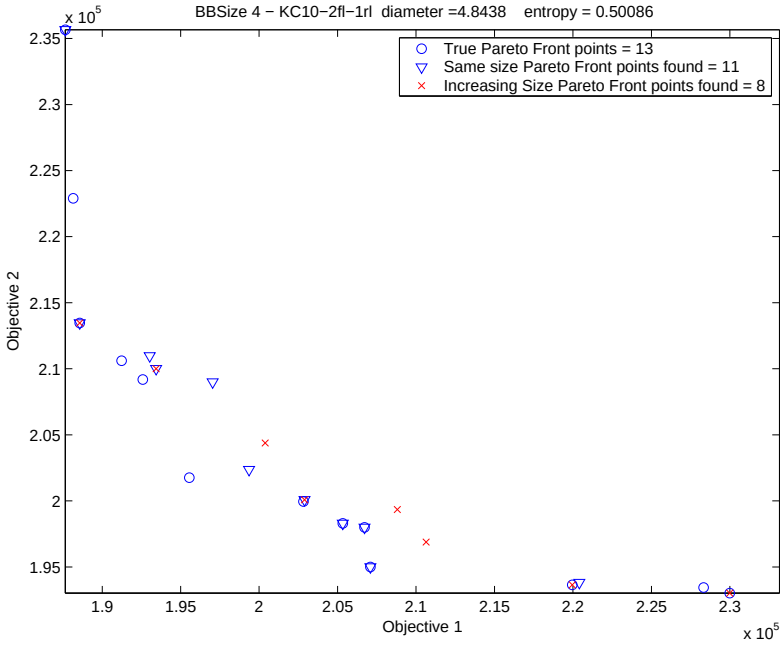

Figure 7.14 Comparison of the effectiveness of using an inherited competitive template BB size 4 vs. initially randomized templates

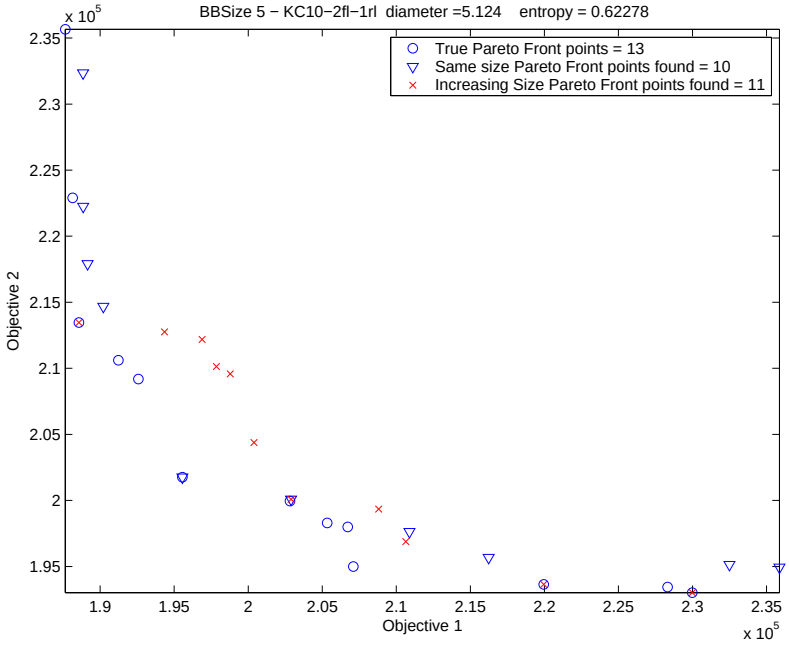

Figure 7.15 Comparison of the effectiveness of using an inherited competitive template BB size 5 vs. initially randomized templates

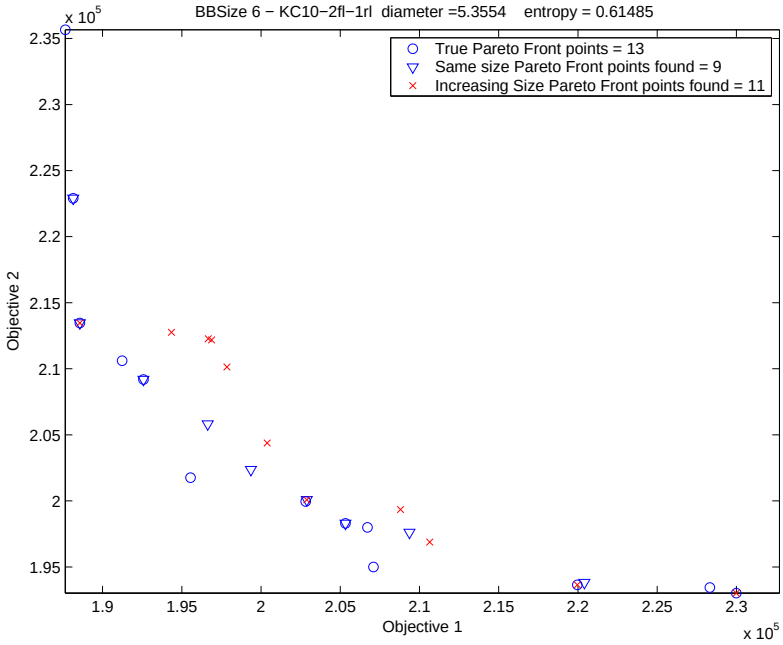

Figure 7.16 Comparison of the effectiveness of using an inherited competitive template BB size 6 vs. initially randomized templates

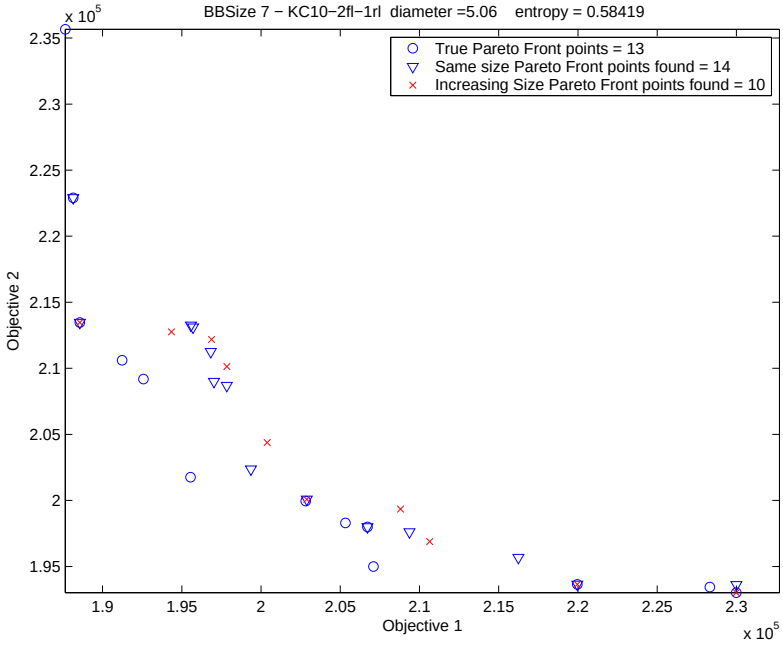

Figure 7.17 Comparison of the effectiveness of using an inherited competitive template BB size 7 vs. initially randomized templates

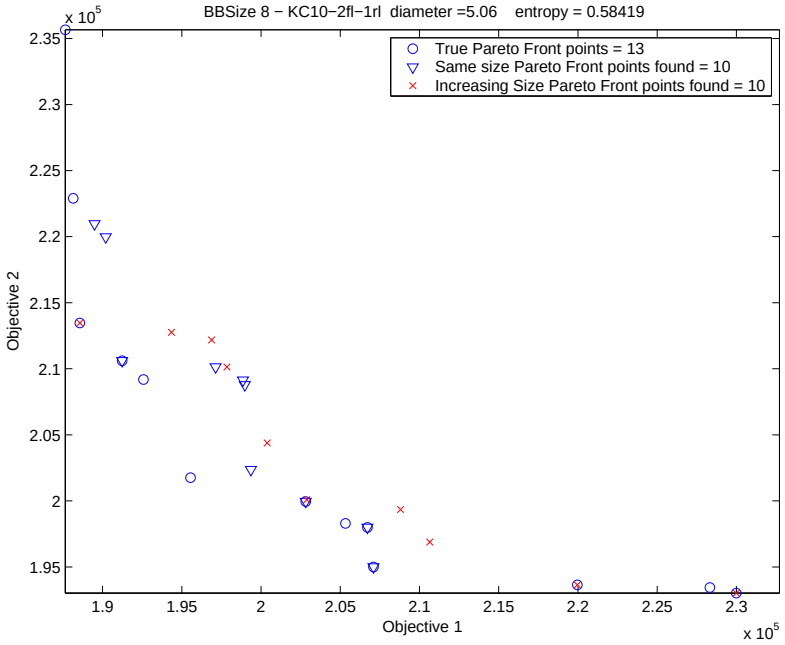

Figure 7.18 Comparison of the effectiveness of using an inherited competitive template BB size 8 vs. initially randomized templates

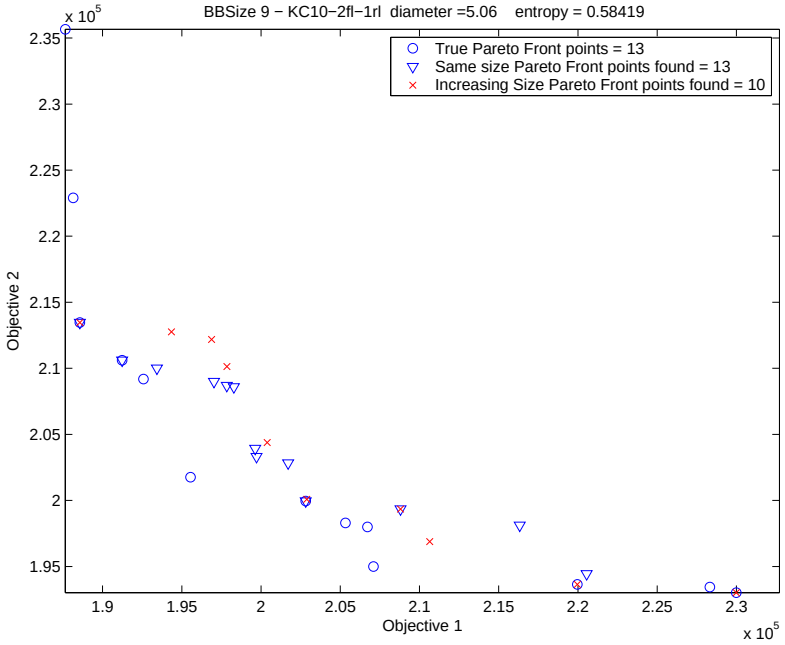

Figure 7.19 Comparison of the effectiveness of using an inherited competitive template BB size 9 vs. initially randomized templates

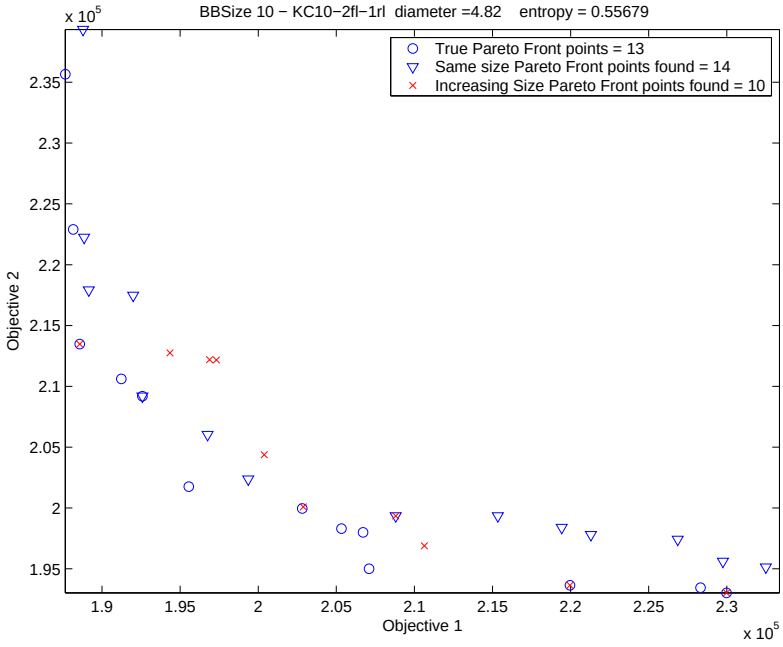

Figure 7.20 Comparison of the effectiveness of using an inherited competitive template BB size 10 vs. initially randomized templates

8. Conclusions and Recommendations

8.1 Introduction

This chapter discusses the conclusions derived from this research. The research goal of creating an effective stochastic algorithm to solve the UAV communication propagation is complete. Section 8.2 lists the conclusions that are based upon the results and analysis in chapter 7. Section 8.3 discusses the future work that can be done to extend this research.

8.2 Conclusions

The conclusions can be decomposed into three main categories, effective algorithm conclusions, building block size conclusions, and competitive template conclusions.

This work fits in well with the current work funded by DARPA. The Defense Department and the Air Force view UAVs as a prominent fixture in future warfare. Some of the current research is looking at multiple UAVs in formations. This research ties in well with that research. Currently, no other research has been published that attempts to optimize a UAV formation based on limiting communication propagation rates. This research is especially useful to the sponsors, the Air Vehicles Directorate and the Information Directorate of AFRL. This research ties into their UAV and simulation research.

8.2.1 Effective Algorithm Conclusions. This investigation attempts to find an effective algorithm to limit the propagation time delay for a formation of heterogeneous UAVs flying in a formation. We found that the MOMGA-II algorithm can accomplish our goal. When the competitive template is randomized between building block sizes, we are able to produce "very good" results as shown in Section 7.4.1. These mean results are nearly twice that of the baseline results, when comparing problems with more than one point on the Pareto Front.

8.2.2 Building Block Size Conclusions. This investigation attempts to see if building block size affects where the individual falls along the Pareto Front. The results show that larger building block sizes tend to spread their search further out on the Pareto Front. The Building Block Location Metric shows that for the instances tested, the larger

building block sizes are twice as likely to populate the outer portions of the Pareto Front than the smaller building block sizes. Section 7.4.2 contains the details and analysis of this experiment.

8.2.3 Competitive Template Conclusions. The goal of these experiments is to determine which method of running the MOMGA-II algorithm is most effective. The research effort found that by randomizing the competitive template between building block sizes, we are able to encounter much better results than if we propagated our competitive template to future building block sizes. The results show that nearly twice as many points are found on Pareto Fronts that consist of more than one point. This is due to the competitive template pulling the search out toward the ends of the Pareto Front rather than pulling it down toward the middle. Section 7.4.3 contains the analysis that support these statements.

8.3 Future Work

While doing this research many new possibilities for future research were uncovered. Sections 8.3.1 and 8.3.2 describe two of these research possibilities.

8.3.1 Chromosome length. In this research, each individual is a binary encoding of 10 digits. In a 10 UAV problem, this equates 100 digits for the entire chromosome. But the chromosome length can be reduced by using the binary representation of decimal numbers, where the largest number would be 1010. By using this representation, the full chromosome length would be reduced to 40 digits. This greatly reduces the search space by 60%. But in doing so, many constraints must be undertaken. First, the researcher must ensure that every binary number generated for the permutation is the binary equivalent of a number between 1 to 10. Plus, he must also ensure that either the chromosome contains no duplicates, or he must handle the case where there are duplicates and assign new values to the duplicated members. This part of the problem is not too difficult. The difficulties arise when trying to use the competitive template to fill in values for underspecified chromosomes. If the competitive template's values match some of the values in the chromosome, a repair mechanism must be implemented in order to correct this

situation. Several approaches are possible to accomplish this, some possibilities are listed below:

- Have multiple competitive templates to choose from
- Replace duplicated UAV with one not currently listed
- Keep competitive template values and replace duplicated values found in the under-specified chromosome.

Each one of these choices has ramifications that may affect the usefulness of the competitive template or the ability of the algorithm to explore the entire search space.

8.3.2 Number of competitive templates. This research effort used one competitive template for every objective. But by looking at the results of the propagating template, this may hinder the algorithm from reaching the center points of the Pareto Front because they are guiding the search toward the outside regions of the Front.

If more competitive templates were placed throughout the Pareto Front, it should pull more values toward the center of the Pareto Front. The difficulty with this approach is finding a good placement of the new templates in order to ensure good spacing along the Pareto Front. Currently, the two competitive templates are based on the best values found for each objective. Any other competitive templates that are added should include a distance function to ensure enough spacing is between it and the other competitive template values.

8.4 Summary

This chapter discusses the conclusions that are uncovered in this research using the mQAP and MOMGA-II. It was found that the MOMGA-II is an effective algorithm for the problem. It was also discovered that larger building block sizes more often populate the outer edges of the Pareto Front. Another conclusion showed that a randomized competitive template at the start of each building block size greatly enhances the search. This chapter also mentions two items to investigate in the future. One looks at reducing the chromosome length while the other increases the number of competitive templates.

Appendix A. Unmanned Aerial Vehicle Introduction

A.1 The Current State of UAVs

UAVs are not new entities. The military has been experimenting with drones for some time now. There have been 11 UAVs developed by the Department of Defense (DoD) since 1964, with 3 of them entering the production phase [97]. In the 1990's, the DoD goal was to have UAVs developed for three different tiers of missions: close range (0 - 50 km), short range (51 - 200 km), and endurance (201+ km) [97]. Later, they combined close range and short range to make the class called Tactical UAV, making only two classes of UAVs. The rest of this section will briefly describe some of the features of the most popular UAVs.

A.1.1 Pioneer. The Pioneer is a short range UAV and a is member of the Tactical UAV family. It was initially procured by the Navy in 1985 to be used for imagery intelligence (IMINT) [97]. It was was used in over 300 missions in the Persian Gulf during 1990 and 1991, and it has been used in Bosnia, Haiti, and Somalia [97]. It flies at a cruise speed of 52 knots and a max speed of 110 knots, can fly as high as 12,000 ft., and it has a range of around 185 km [97]. It is made by Pioneer UAV, Inc., a joint U.S./Israeli venture, and as of 1998 they were still a part of the Naval inventory [97].

A.1.2 Hunter. The Hunter is another short range UAV in the Tactical UAV class. It was an Israeli made UAV that the Army decided to purchase. It flies at speeds of 70 knots (cruise) up to 110 knots (max), it could fly as 16,000 feet, and has a range of 125 km [97]. The program was cancelled in 1996 after cost overruns, schedule slippages, and 20 vehicle crashes [97].

A.1.3 Pointer. The Pointer is a hand-launched close range UAV in the Tactical UAV class. It is easily assembled and only weighs 8.5 pounds. It's range is around 5 to 7 km and can reach altitudes of 3000 feet [97]. The missions usually last one hour or less and the UAVs posses no autonomous control [97]. This UAV is a Naval procurement that has it's limitations, due to its size and payload capacity, but in 1999 they were fitted with infrared red cameras that now enable it to do night missions [97].

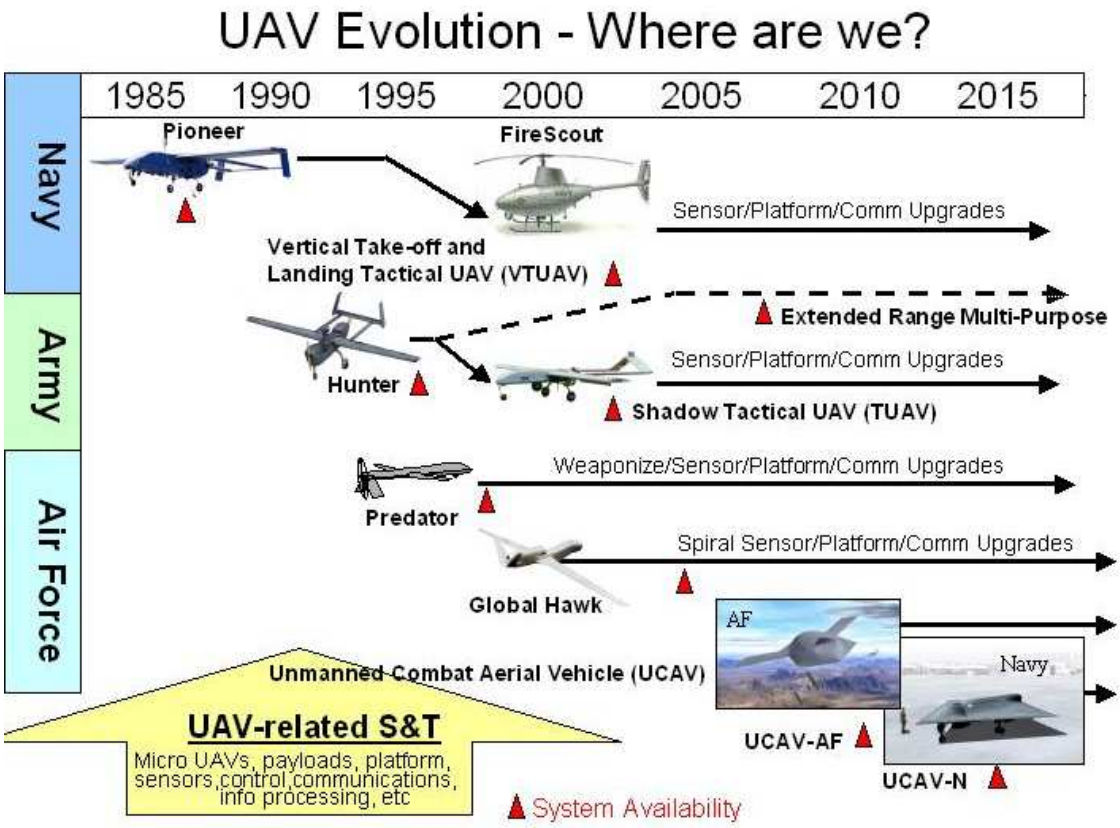

Figure A.1 UAV Timeline
[97]

Table A.1 **Current UAV Particulars** *This table contains a partial listing of some of the current UAVs and listing their stated endurance, payload weight, and altitude limitations .*

UAV Name	Endurance (hrs)	Payload (lbs)	Altitude (ft)
DarkStar	8	1000	45,000
Eagle Eye	8	300	20,000
Exdrone	2.5	25	10,000
Firebee	1.25	470	60,000
Guardian	6.25	220	18,000
Global Hawk	42	1960	65,000
Gnat 750	48	140	25,000
Hunter	12.	200	43,000
Model 324	2.5	200	43,000
Model 410	12	300	30,000
Outrider	4	160	15,000
Pioneer	5.5	75	12,000
Pointer	1	2	3,000
Predator	29	700	+40,000

[97]

Figure A.2 The Pioneer UAV
[97]

Figure A.3 Hunter
[97]

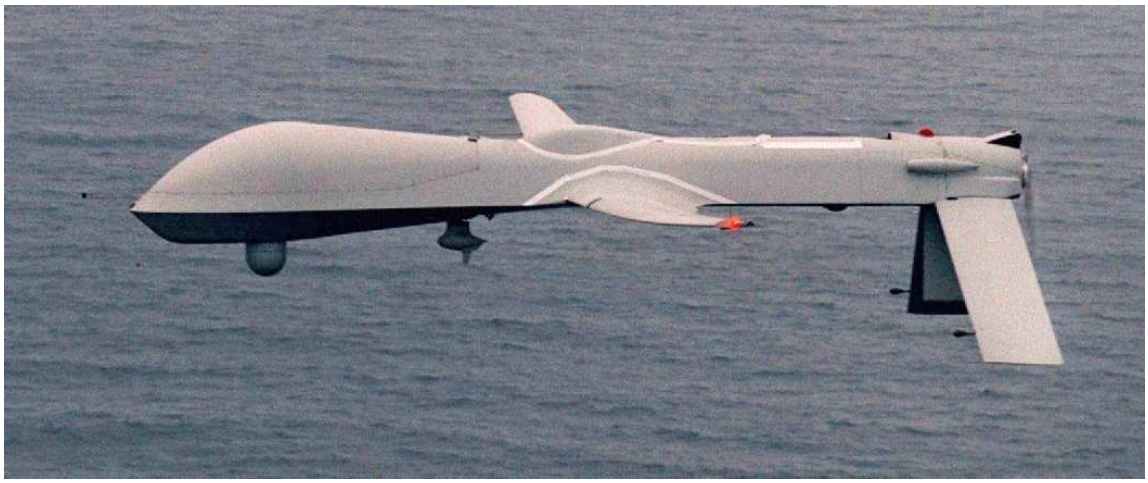

Figure A.4 Predator
[97]

A.1.4 Predator. The Predator is a Medium Altitude Endurance (MAE) UAV and a member of the Endurance class of UAVs. This UAV has become well known recently because of the CIA arming one with a Hellfire missile and using it to kill six al-Qaida members [97]. The Air Force is the lead service of the Predator, but the CIA uses it in their operations as well. The Predator needs a 5000 x 125 foot runway and requires line-of-sight with its ground control station but unlimited with satellite usage [97]. The Predator is capable of speeds of 90 knots (cruise) to 120 knots (Max), has an unlimited range (via satellite), and can fly as high as 26,000 feet [97]. It was used extensively Bosnia, Kosovo, and Afghanistan, flying over 600 missions for NATO, UN, and US forces [97]. Figure 12 shows a diagram of the predator's usage in the combat arena, showing its communication links and its sensor footprint.

A.1.5 Guardian. The Guardian is a vertical takeoff and landing (VTOL) UAV that is a member of the Tactical UAV group. It is manufactured by Canadair for the Navy [97]. Its maximum speed is 85 knots, can fly as high as 18,000 feet, and has a range of 200 km [97]. The Guardian benefits from the lessons learned from its precursor, the Sentinel, which was developed over the course of 10 years [97]. This UAV is well suited for use off of ships and in rough terrain, where runways are impractical [97].

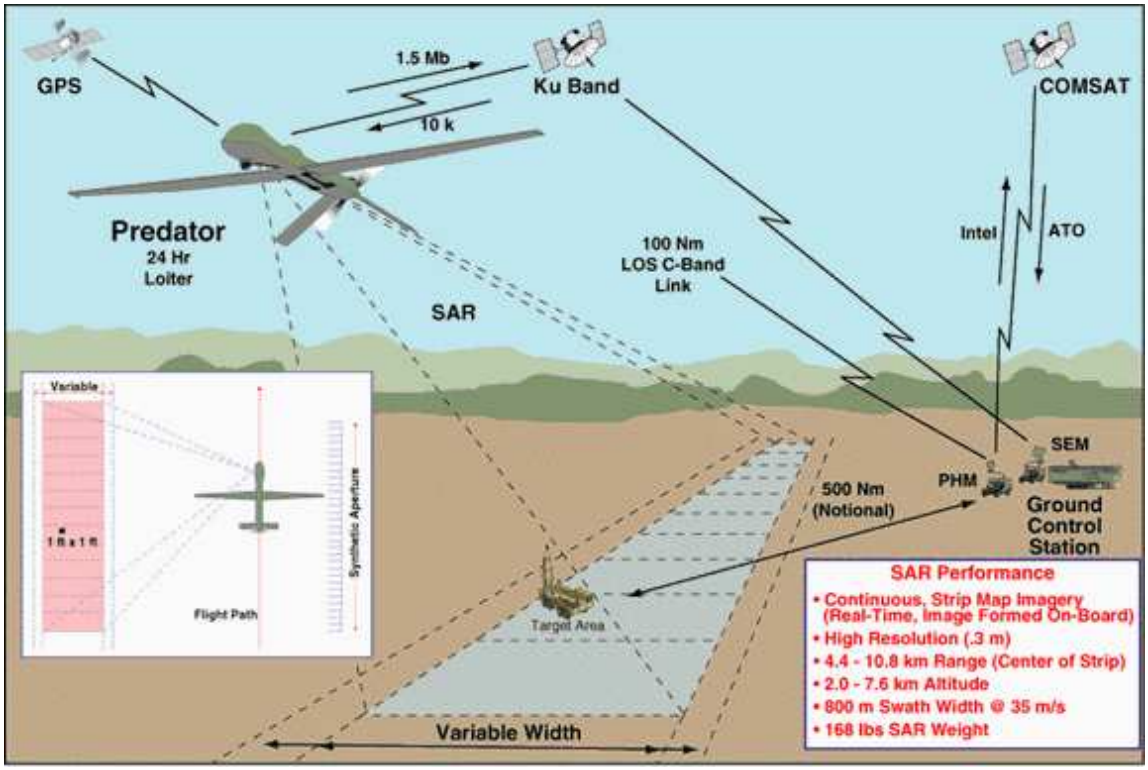

Figure A.5 The Predator's SAR Image
[97]

Figure A.6 Guardian
[97]

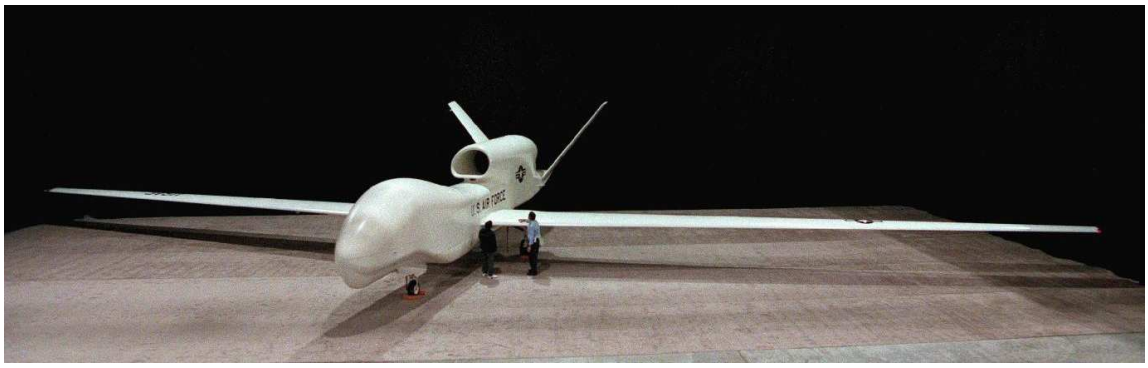

Figure A.7 Global Hawk
[97]

A.1.6 Global Hawk. The Global Hawk is a High Altitude Endurance (HAE) UAV and is a member of the Endurance class of UAVs. Northrop Grumman was awarded a contract in 2000 to provide two prototype vehicles before March 31, 2002. But the Air Force called them into service earlier than planned, due to the terrorist attacks on September 11, 2001, and on June 15, 2002, the system reached 1000 combat flying hours [108]. The Global Hawk can fly at a cruise speed of 343 knots, can fly at 65,000 feet, and has a range of 3000 nautical miles [97] [33]. It is capable of covering 40,000 square nautical miles in a day with a resolution of three feet (one foot spot resolution when needed) [33].

A.1.7 Outrider. The Outrider UAV is a short range UAV and a member of the Tactical UAV group. Alliant Techsystems was awarded the contract to deliver the UAV to the Army, Navy, and Marines [97]. It is being developed to replace the Pioneer UAV [97]. The Outrider is capable of a cruise speed of 90 knots with a maximum speed of 110 knots, it can attain altitudes of 15,000 feet, and has a range of 200 km [97].

A.1.8 Darkstar. The Darkstar UAV is in the Endurance class of UAVs. It was a stealthy designed by Lockheed Martin and Boeing. It was supposed to have a cruise speed of 300 knots, an altitude of 50,000 feet, and a range similar to that of the Predator [97]. Unfortunately, the Darkstar met its demise in 1999. It was determined that it was no where close to being used as an operational system and the Air Force deemed the redesign costs were too much, so the program was cancelled [97]. It is included in this paper because

Figure A.8 Outrider
[97]

it represents a unique design that may be revisited in the future because of its stealth characteristics.

A.2 The future of UAVs

A.2.1 Micro Aerial Vehicles (MAV). Micro Aerial Vehicles were introduced by DARPA in a 1992 meeting. The idea was met with some scepticism, but began to pick up steam in 1994, and in 1997 DARPA initiated a $35 million dollar development program [97]. DARPA wants a microdrone with the characteristics outlined in table 2. The goal is to use these UAVs for battle damage assessment, artillery spotting, sensor dispersal, communications relay, detecting mines and hazardous materials, and radar jamming [97]. MAVs capable of vertical flight could be used to scout buildings, counter-terrorism operations, and could be supplied with pilots so if they are downed, the pilots could launch the MAVs to do surveillance or relay communications to search and rescue parties [97]. Several MAVs have been developed through the DARPA effort [97], and these efforts are being investigated by other countries.

A.2.2 The X-45A. Currently, Boeing is working on an aircraft called the X-45A [115] [97]. This aircraft is being designed explicitly for the purpose the suppression of

Figure A.9 DarkStar
[97]

Table A.2 **Desired MAV Characteristics**

Characteristic	DARPA Goal
Size	6 inches
Weight	4 ounces
Endurance	2 hours
Range	6 miles
Control	Autonomous
Max Winds	30 MPH
Sensor	Day/Night Camera
Cost	less than $1000

[97]

NASA Dryden Flight Research Center Photo Collection
http://www.dfrc.nasa.gov/gallery/photo/index.html
NASA Photo: ED01-0209-5 Date: July 14, 2001 Photo by: Nick Galante/PMRF
The Helios Prototype flying wing is shown near the Hawaiian islands of Niihau and Lehua during its first test flight on solar power from the U.S. Navy's Pacific Missile Range Facility.

Figure A.10 The Helios Solar Powered Wing
[97]

enemy air defenses (SEAD), but some studies are looking into their use for peace-keeping patrols [115][97]. The SEAD task involves flying around a combat area looking for missile launchers to switch on their radar. When a radar is detected, the aircraft fires a homing missile to take out the radar and the missile site [115]. For this dangerous task, it would be very beneficial to use unmanned aircraft. This is considered the second most difficult task to do, behind aerial dog fighting [115]. Boeing's goal is to have these UCAVs enter the service inventory by 2010 [115]. Boeing wants to reduce the data traffic between the operator and the aircraft, so they are looking for ways to make the aircraft more self-reliant [115]. Embedding swarm characteristics into each aircraft would help to make the aircraft more autonomous.

Figure A.11 X-45A
[97]

Appendix B. Parallel Programming

B.1 Parallel MOEAs

There are some very hard real-world problems that can take a GA a very long time to converge to an answer. In order to speed up the process, a parallel implementation of the GA may be used. GAs are naturally parallelizable because they maintain a population of solutions [35]. There are three main ways that GAs can be parallelized. These include asynchronous, course-grained, and fine-grained.

In a course-grained PGA the population is structured as a stepping stone population model. The parallelism is obtained by the parallel execution of a number of GAs operating on one of the subpopulations. Communication occurs occasionally so that the processors can exchange migrating individuals. These are most efficiently implemented on MIMD systems [54].

Parallelism is limited due to the fact that each step operates on a subpopulation. The scalability is good and it is a straightforward way to parallelize a GA.

For the fine-grained model the population is structured in an isolation-by-distance model. The population is mapped onto a grid and a neighborhood structure is defined for all of the grid points. Selection and crossover are restricted to the neighborhood. Parallelism occurs when you assign a process to each individual. Communications is only necessary during selection and crossover.

This model is best suited for SIMD systems because each process operates on only one individual and communication can be synchronized easily. These also offer a maximum amount of parallelism. Each individual can be evaluated and mutated in parallel. And the scalability is also very high [54]

B.2 General Parallelization Concepts of the Problem

B.2.1 Why its Interesting. Parallelizing this problem can increase the efficiency and/or effectiveness. By parallelizing the problem, more processing power works on the same problem. And if communication time is limited, the faster the program arrives at its

solution, the better. This is one measurement of efficiency. Another measure of efficiency is the amount of processing power that is in use. If there is poor load balancing, the efficiency of the total processing power is less than if an effective load balancing scheme is used and each processor is kept busy.

Effectiveness is a measure of how "good" a solution is. For example, a deterministic algorithm is very effective, but may not be very efficient. Likewise, a stochastic algorithm can be very efficient but not very effective. The goal is to get a balance of between efficiency and effectiveness.

There are three major parallel paradigms that need to be looked at in order to determine which one is best for this problem. These paradigms are:

- Master-Slave Model
- Island Model
- Diffusion Model

B.2.1.1 Master-Slave Model. The Master-Slave model is implemented in an MOEA by having the objective functions distributed among several slave processors and the master processors takes care of all of the other operations and overhead. Figure B.1 shows a diagram of how the Master-Slave model is implemented. For this model the number of processors in use is independent of the solutions are evaluated. [21] The master processor's main job is to control the parallelization of the objective function evaluations. Communication will occur between the master and slave processors at the end of each generation, so cost is generally derived from the number of slaves and the hardware platform. [21] This implies that the Master-Slave model is generally more efficient as the objective evaluations become more expensive, provided the computational loads are spread evenly among the slaves. This method's main usefulness is in increasing the efficiency of the problem [133]. For the application in this project, the objective functions are not very expensive, so this model might not be as efficient for this application versus other applications. This model should provide near linear speedup [53, 79]. Normally, the Island model is used on coarse grained or MIMD architectures [85].

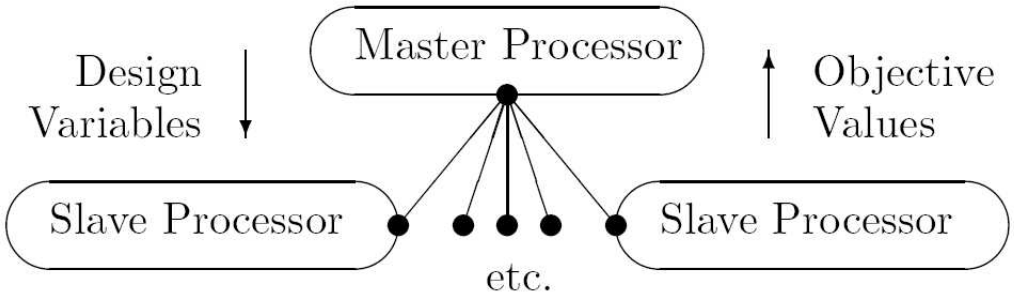

Figure B.1 Master-Slave Model

B.2.1.2 Island Model. The Island model is used in an MOEA to develop subpopulations in parallel and then occasionally migrate some of the population members from one island to another. Figure B.2 shows a diagram of the model. The evolution of subpopulations allows for exploration of the search space, while the migration of population members aids in the exploitation of the search space. The amount of exploitation varies based on the migration policy defined. For example, migration occurring at the end of each generation causes divergence toward a solution much faster than migrating after every ten generations. Another important factor is the number and type of individuals being migrated between the islands. This implementation is also known as a *course-grained* implementation because each island contains a number of individuals [21]. This method is used to improve effectiveness or efficiency.

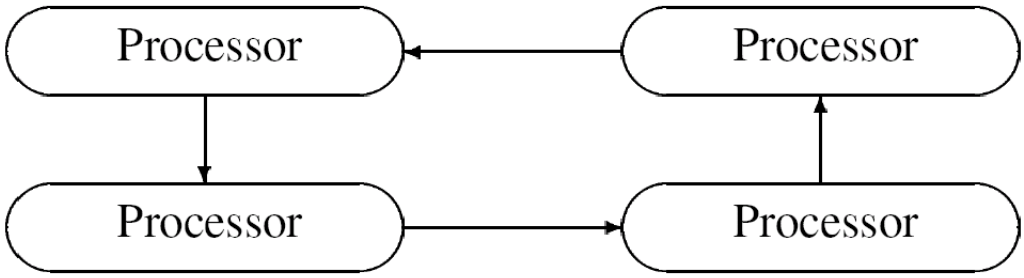

Figure B.2 Island Model

B.2.1.3 Diffusion Model. The diffusion model deals with one population, where each processor holds a small number of individuals. Since only one population is involved, this is sometimes referred to as a *fine-grained* implementation. Figure B.3 shows

a graphical depiction of the diffusion model. This model parallelizes the EA itself, so the communication cost can be very high. This is why this implementation is most often used with shared memory processors (SMP). Because of the high communication rate needed for this method, it was decided not to go with this one since communication costs can be rather high.

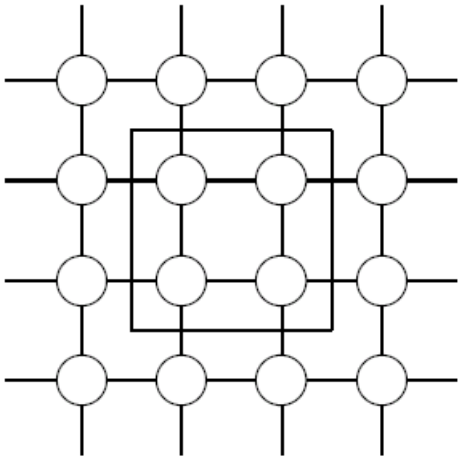

Figure B.3 Diffusion Model

B.2.2 Data Structure Decomposition. One of the key things to decide when solving a problem in parallel is how to break-up the computations. This is called decomposing the problem. This section describes briefly some of the more common approaches to problem decomposition.

B.2.2.1 Recursive Decomposition. "Recursive decomposition is a method for inducing concurrency in problems that can be solved using the divide and conquer strategy" [51]. Basically, this approach keeps dividing the data into smaller sections until the smallest possible representation of the problem. Then it returns the answers back up the chain. The approach is a rather simple approach and below is an algorithm that shows the basic structure of its implementation:

Step 1) Is the problem at the lowest possible representation?

If yes, return result.

If no, go to step 2.

Step 2) Divide data in half.

Step 3) Send both sets of data back to step 1.

As the results are returned back to the caller, the function calculates its the results of its calculation and returns it up the chain until the answer is output. An example of this method in use is the quicksort algorithm

B.2.2.2 Data Decomposition. Data decomposition is a good method to use when deriving concurrency in algorithms that operate on large data structures [51]. The decomposition is done in two steps, as seen below:

Step 1) Partition the data on which the computations are to be performed.

Step 2) Partition the computations into tasks.

This partitioning can be done in many ways. The user need to look at the data and decide the best way to decompose the data in a natural and efficient computational decomposition [51]. An example of this type of decomposition is matrix-multiplication. In fact, most matrix operations can be formulated using data decomposition.

B.2.2.3 Exploratory Decomposition. Exploratory decomposition is best for decomposing problems that the computations are used to search a solution space. For this decomposition, the search space is partitioned into smaller parts and the search is done concurrently in each one of these smaller spaces until the desired solution is found.

This is different than data decomposition because this method will terminate early if a solution is found, while data decomposition runs all tasks to completion. This means that the work performed in the parallel implementation can be very different from the serial implementation.

B.2.2.4 Speculative Decomposition. Speculative decomposition is a method used when a program has the choice of taking one of many computationally significant branches depending on the output of other computations that precede it [51]. What happens is that while one task is performing the computation to determine the next computation, the other tasks are concurrently starting the computations for that next stage. Then when the first task finishes its calculations and the next stage is determined, the only results used are from the task that performed the calculations for the chosen next stage. All other run tasks are discarded. While this seems like wasted computation, it actually speeds up the process. Because if these were done in serial to begin with, all of the other processes would be idle, waiting to be used. But this allows them to do some work that in the end is useful because it does the calculations for the next stage ahead of time and does it in parallel with the previous stage, rather than serially.

B.2.2.5 Hybrid Decomposition. A hybrid decomposition is simply a technique that combines two or more decomposition techniques. This is especially useful when different types of decomposition are needed in different stages of the computation.

B.2.2.6 Speedup. The goal of parallelizing an application is to improve its wall clock time. But how much better does the parallel program do when compared to the best known serial implementation? The speedup equation captures the benefit of solving a problem in parallel vs. serially. The formal definition of speedup, S, is the ratio of the serial runtime of the best sequential algorithm to the time taken by the parallel algorithm to the same problem on p processing elements. Equation B.1 is the equation for speedup

$$Speedup = \frac{T_s}{T_p} \tag{B.1}$$

where T_s is the fastest known time to solve the problem in a serial manner and T_p is the time it takes to solve the problem in a parallel manner. The reason why the best serial implementation is chosen is because there are some serial algorithms that cannot be parallelized, so to not include them would give skewed results toward parallelization [51].

B.2.2.7 Efficiency. Speedup can only show the relative improvement of an application based on time. It cannot tell you what percentage of the processing element are being used by the application. Efficiency is the measurement that can do that. By definition efficiency is the amount of the time that the processing element is gainfully employed, which is calculated by using the ratio of the speedup value and the number of processing elements, as shown in equation B.2.

$$E = \frac{S}{p} \tag{B.2}$$

where E is the efficiency, S is the speedup, and p is the number of processors used. This can be broken down into the function in equation B.3.

$$E = \frac{T_S}{pT_P} \tag{B.3}$$

It can also be written based on the overhead time, as shown in equation B.4.

$$E = \frac{1}{1 + \frac{T_o}{T_S}} \tag{B.4}$$

where the overhead function is defined as B.5:

$$T_o = pT_P - T_S \tag{B.5}$$

B.2.2.8 Work. Work is defined as the product of the parallel runtime and the number of processors used, as shown in equation B.6.

$$W = pT_p \tag{B.6}$$

This is also known as the cost of a system. A parallel system is said to be cost-optimal if the cost of solving a problem on a parallel system has the same asymptotic growth of the fastest known sequential algorithm, which means $W = pT_P = T_S$. Cost optimality is

a very important practical concept, because even small differences in cost optimality can have a huge effect on the scaling of the application [51].

B.2.2.9 Scalability. ***Isoefficiency*** The isoefficiency determines the ease with which a parallel system maintains a constant efficiency. This means that it achieves speedups that increase in proportion to the number of processors. A small isoefficiency function is good. This means that small increments in the problem size are sufficient for the efficient utilization of an increasing number of processors. Basically this implies that the parallel system is highly scalable. Likewise a large isoefficiency value indicates a poorly scalable system. The equation for isoefficiency is generally written as equation B.7 or equation B.8.

$$W = fracE1 - ET_o(W, p) \tag{B.7}$$

$$W = KT_o(W, p) \tag{B.8}$$

where $K = E/(1 - E)$.

Amdahl's Law But the metrics described earlier can be used to analyze and predict the scalability of various parallel combinations. One such performance model is Amdahl's law. This performance model is based on a fixed problem size. Equation B.9 shows Amdahl's Law as defined in [59]

$$S_n = \frac{W}{\alpha W + (1 - \alpha)(W/n)} = \frac{n}{1 + (n - 1)\alpha} \rightarrow \frac{1}{\alpha} \tag{B.9}$$

$$as\, n \rightarrow \infty$$

where W is the workload, α is the percentage of the workload that must be executed simultaneously, and n is the number of nodes. This is one of the most fundamental laws when studying parallel systems. Some of the implications of the law include [59]:

1) The sequentially executed part of the code is a bottleneck,

2) In order to get good speedup, the sequential bottleneck, α needs to be as small as possible,

3) The common case should be optimized.

When incorporated with the inherent parallel overhead, the speedup equation is changed to

$$\begin{aligned} S_n &= \frac{W}{\alpha W + (1-\alpha)(W/n) + T_o} \\ &= \frac{n}{1 + (n-1)\alpha + \frac{nT_o}{W}} \\ &\rightarrow \frac{1}{\alpha + \frac{T_o}{W}} \\ &as\ n \rightarrow \infty \end{aligned} \tag{B.10}$$

This equation shows that in addition to reducing the sequential bottleneck, the average granularity must increase in order to reduce the bad impact that that overhead has to the performance of the code [59]. This is considered a conservative approach.

Gustafson's Law Another performance model is Gustafson's Law. It alleviates the the sequential bottleneck by using a fixed-time concept instead of a fixed problem size. The fixed-time method scales the problem size with the increase in machine size, thus keeping the time it takes to finish the problem fixed. The formula for Gustafson's Law is in equation B.11.

$$\begin{aligned} S'_n &= \frac{Sequential\,time\,for\,scaled - upworkload}{Parallel\,timed\,for\,scaled - up\,workload} \\ &= \frac{\alpha W + (1-\alpha)nW}{W} \\ &= \alpha + (1-\alpha)n \end{aligned} \tag{B.11}$$

This states that the fixed-time speedup is a linear function of n, if the workload is scaled up to maintain a fixed execution time [59]. Equation B.12 shows Gustafson's law with the overhead added in.

$$S'_n = \frac{\alpha W + (1-\alpha)nW}{W + T_o} \tag{B.12}$$

$$= \frac{\alpha + (1-\alpha)n}{1 + T_o/W} \tag{B.13}$$

Sun and Ni's Law Another performance measure is one that is a memory bounded speed-up model. This model generalizes Amdahl's and Gustafson's Law in order to maximize the use of the CPU and memory capacity [59].

$$S^*_n = \frac{Sequential\,time\,for\,scaled\,workload}{Parallel\,timed\,for\,scaled\,workload} \tag{B.14}$$

$$= \frac{\alpha W + (1-\alpha)G(n)W}{\alpha W + (1-\alpha)G(n)W/n}$$

$$= \frac{\alpha + (1-\alpha)G(n)}{\alpha + (1-\alpha)G(n)/n}$$

With overhead included the equation is as follows:

$$S^*_n = \frac{\alpha W + (1-\alpha)G(n)W}{\alpha W + (1-\alpha)G(n)W/n + T_o} \tag{B.15}$$

$$= \frac{\alpha + (1-\alpha)G(n)}{\alpha + (1-\alpha)G(n)/n + T_o/W}$$

This equation has three special cases [59]:

1) When $G(n) = 1$, the equation becomes Ahmdahl's equation.

2) When $G(n) = n$, the equation becomes Gustafson's equation.

3) When $G(n) > n$, the equation is Sun and Ni's memory-bound equation.

Appendix C. Messy Genetic Algorithms

C.1 Problems with GAs

Genetic algorithms have been successfully applied in many applications - commerce, search, optimization, and engineering to name a few. While GAs are a valuable tool for researchers, there are some problems with this algorithm. Some of the problems include the following [50, 63]:

- The relation, class, and sample spaces are combined. So decision making in each space affects the other two. Therefore the overall decision making process is noisy and susceptible to error.
- No precise mechanism for implicit parallelism occurs.
- Only a poor search for relations can occur. Since only a small sample of the total number of relations can be defined, crossover and mutation are used to search for other relations. A problem exists when there are two bit values that are that are tightly linked and they are on opposite ends of the gene. These are likely to be disrupted through crossover. This is defined as a *linkage problem*.

Another problem for GAs is known as the *deception problem*. This problem occurs when a GA is used to solve a hard multimodal optimization problem [50, 68]. This occurs when a number of *deceptive subfunctions* mislead the GA to cause it to converge to a locally optimal point instead of a global optimal. An example of this occurs when two schema with high fitness values (11xxx and xxx00) are combined (11x00) to give a lower fitness value of its complement (00x11).

C.2 mGA Differences

Messy GAs attempt to overcome some of these problems by changing some of the coding rules and the operators. The key differences between the mGA and the standard GA are outlined in table C.1 [29].

Table C.1 **Main differences between a standard GA and a mGA**

	Standard GA	Messy GA
String length	Fixed length	Variable length
Overspecification allowed?	No	Yes
Underspecification allowed?	No	Yes
Competitive Template used?	No	Yes
Main Operator Used	Crossover	Cut and Splice
Phases in evolutionary process	1	2

C.3 Messy encoding

A GA uses fixed sting lengths for a gene and each allele in the gene has a defined characteristic it represents. But a mGA uses an ordered pair encoding system that includes both the value of an allele and its location in the gene. This type of encoding overcomes the *linkage problem* described earlier.

Kargupta [63] has formally denoted the mGA representation in equation C.1:

$$I_m : \chi \rightarrow S_l \times \Lambda^l \tag{C.1}$$

where I_m is the mGA individual, l is the length of the chromosome and S_l is the set of all possible permutations of the chromosome locations. Also note that χ can be represented by $l!$ different possibilities. This allows related genes to be clustered together during a search and limit the disruption that operators can have on good building blocks.

Since the messy encoding includes the location of each allele, it allows for overspecification and underspecification of the gene. This means that there can be multiple values represented for a location (overspecification) or there can be no values listed for a location (underspecification). Because of this implementation, there must be defined a way to assign one value to every location. With overspecification a user could pick the first value listed, the last value listed, a random value, or some other sort of method [29, 48, 50, 63]. The most common method is to pick the first value listed. For underspecification, a competitive template is used to fill in any missing variables. The competitive template is a fixed length representation of the gene with values filled in for every location. The values

are usually assigned to the template based on the fitness value of a gene from a previous generation.

C.4 mGA Operators

The mGA generally utilizes three basic operators, a selection operator, a version of the crossover operator called cut and slice, and an allelic mutation operator.

C.4.1 Selection Operator. The selection operator is one of the key operators used in evolutionary algorithms. The goal is to use a selection operator that promotes better population members. It's purpose is to select the better population members and reject the population members that are not as good. While any number of selection operators can be used, there have been two selection operators that have been predominately used with the mGA, tournament selection and thresholding selection.

Tournament selection is a method in which a group of q individuals are chosen from the population and the one with the highest fitness value is selected. This process is repeated until the allotted number of individuals has been chosen for the new population. The members of the tournament can be chosen with or without replacement from the population. While a tournament size of $q = 2$ is quite common, any number of q individuals can be chosen from the population. As the tournament size increases, the selection intensity and loss of diversity increases, while the takeover time decreases. A researcher can experiment with the tournament size in order to get the right balance of exploration and exploitation to meet his needs.

Thresholding selection is used to "ensure that only classes that belong to a particular equivalence relation are compared to one another" [63]. An example of this can be shown with three strings: ((0 1)(1 0)), ((1 0)(0 0)), and ((2 0)(1 1)). The building blocks defined in the three strings are 10#, 00#, and #10 respectively. Comparing the fitness values of the first two strings is good because they are from the same relation, while the third string has a different relation associated with it and is restricted from competing with the other two strings. Thresholding selection allows strings to compete only if their θ is larger than

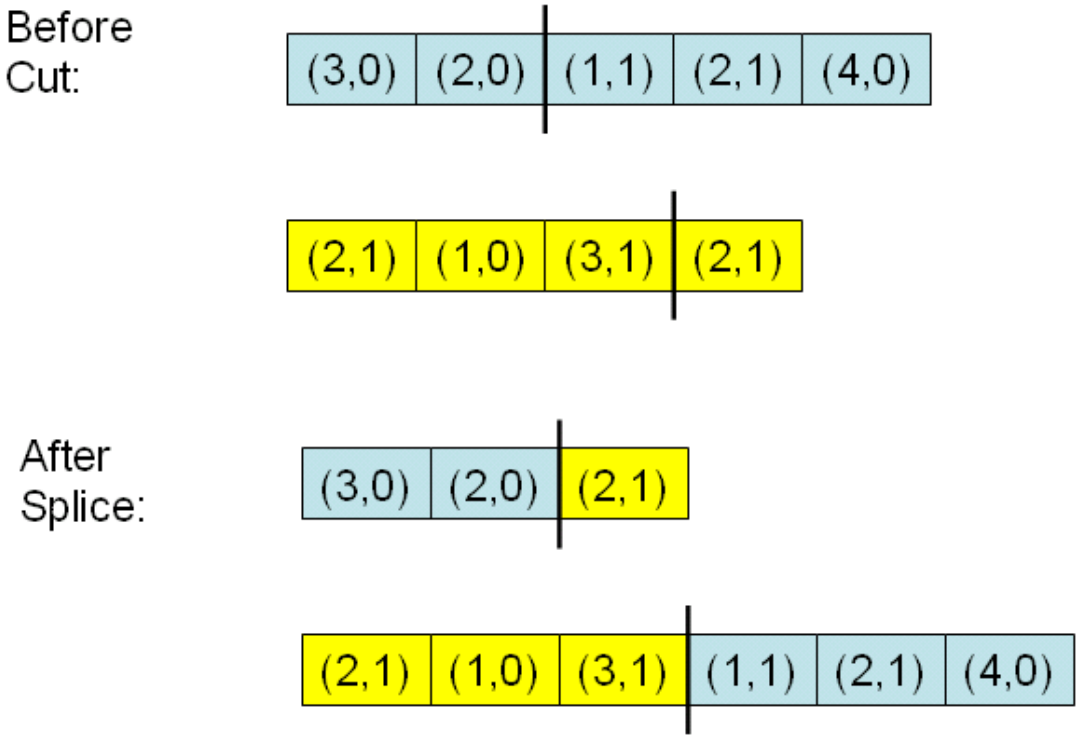

Figure C.1 Example of Cut and Splice operation

a defined threshold value of $\overline{\theta}$. $\overline{\theta}$ is defined as $\overline{\theta} = \frac{l_1 * l_2}{l}$, where l_1 and l_2 are the length of the two strings and l is the problem length.

C.4.2 Cut and Splice Operator. Since the mGA allows strings of variable lengths, the simple crossover operator does not work. To overcome this, the cut and splice operator was conceived. This operator simply cuts two strings and then concatenates the end portion of one string to the front portion of the other string. See figure C.1 for an example of how the cut and splice operator would work.

Goldberg calculates the overall cut probability, p_c, as $p_c = p_\kappa(\lambda - 1)$ where p_κ is a bitwise cut probability, λ is the current string length, and p_c must meet the limitation of $p_c \leq 1$ [50]. This means that small strings have a lower probability of getting cut than the larger strings. If p_κ is set low, then the smaller strings have a much less probability of getting cut than if p_κ is set high.

The splice operation also has a probability of p_s assigned to it. So by adjusting the p_κ and the p_s, a researcher can increase or decrease the amount of time that the mGA takes to explore the search space. Goldberg [50] refers to the cut and splice operations as two separate operations, where I refer to them as one operation. I do this only to show they have a close relationship to each other, unlike mutation or the selection operator which can be changed without regard to the two operations.

Overall, the cut and splice operator have two limiting types of behavior. Since strings are short at the beginning of the run, the p_c is small so the chances of cuts occurring is low, while the splice operation is a constant. This results in more splice operations occurring early in the run. As the run continues, the string lengths get longer due to more splice operations. This causes the p_c to increase. Eventually this creates an operation very similar to one-point crossover [48].

C.4.3 Mutation Operator. The mutation operator is used to randomly change the allele values of the chromosome. The probability of mutation, p_m, is set by the researcher to allow for a diverse pool of allele values. This operator is often considered a background operator [35] and is very beneficial in increasing the exploration of the algorithm.

C.5 mGA Phases

The mGA has two loops, the outer and inner loop. The outer loop is the number of eras that the algorithm runs the inner loop. The outer loop terminates when the user defined stopping criteria are met. The inner loop has three phases: the initialization, primordial, and the juxtapositional phase. The next few sections briefly describe the three phases.

C.5.1 Initialization Phase. For the initialization phase, the mGA uses a technique called partially enumerative initialization (PEI), which provides at least one copy of all possible building blocks that are a specified length. The specified length of the building blocks should be chosen in order to include the highest order of deceptive nonlinearity that can be expected in the problem [45, 50]. By doing this, it can be expected that good

building blocks are combined and create good solutions. But there is a huge downside to having all of these building blocks because each one needs to be evaluated in order to determine the fitness values of each of the building blocks. This results in a population size of $n = 2^k\binom{l}{k}$ where n is the population size, l is the problem size, and k is the order of deceptive nonlinearities. This creates a total of $\binom{l}{k}$ gene combinations that are size k and 2^k different allele combinations for for each gene combination[48].

C.5.2 Primordial Phase. The primordial phase follows the initialization phase. This phase is responsible for selecting the good building blocks and nothing else. The goal is to weed out some of the bad building blocks and decrease the population size before the juxtapositional phase. By doing the selection process at this point in the algorithm, the population contains fewer and better building blocks for the genetic operations that occur in the next phase. The selection operators commonly used in this phase are tournament selection [50] and thresholding selection [63].

C.5.3 Juxtapositional Phase. The juxtapositional phase occurs after the primordial phase. The juxtapositional phase operates much like the processing of a simple GA does. First, selection is used to determine which individuals are chosen to be mated using the cut and splice operator. After the cut and splice operation is implemented, a mutation operator can be used, if desired. According to Goldberg [48], several mutation operators have been defined, but mGA tests never use them. The reason for this is to test the algorithm as stringently as possible and show that it has the capability to exploit the initial population diversity.

C.6 The disadvantages of the mGA algorithm

There are three disadvantages of the mGA [63]:

- Search space decomposition is not sufficient.
- One local search template.
- Implicit parallelism is lacking.

These disadvantages are briefly described in the following sections.

C.6.1 Decomposition. While the mGA does take steps toward decomposing the sample space from the relation space and class space, through the use of the messy representation, it may not be decomposed enough. This is evident when doing comparisons of building blocks that belong to different relations. This creates competition among classes of good relations, which should be avoided. [63]

C.6.2 Local search template. The competitive template of the mGA is originally chosen at random. This can bias the algorithm in such a way that it can make a class of problems either easier or more difficult based on the competitive template that is randomly generated. [63]

C.6.3 Implicit parallelism. The biggest drawback of the mGA algorithm is the lack of implicit parallelism. This results in longer serial running times. The serial time complexity of the mGA was analyzed by Goldberg [49] and table C.2 shows the serial time complexity of the mGA.

Table C.2 **Complexity Estimates for the mGA**[49]

Phase	Serial Complexity Estimate
Initialization	$O(l^k)$
Primordial	O
Juxtapositional	$O(l \log l)$
mGA Total	$O(l^k)$

The complexity analysis illustrates that the Initialization phase is by far the longest running portion of the algorithm. This is the phase that creates the initial population using PEI.

Appendix D. Symbolic Mapping of MOMGA-II to the mQAP

D.1 Problem Domain Requirements Specification

This section converts the mQAP into a symbolic specification. We use a graph model to depict the mQAP. First, the definition of the graph model is formulated.

The input domain is set-up. Basically, the input for this problem are vertices and edges, which represent the locations and distances between the locations, respectively. Mapped onto the locations, are the facilities which have flows going from each facility to the other facilities. There can be more than one category of flow mapped to the facilities, since we are dealing with a mQAP.

The output domain is simply the sequence of the facilities mapped to locations that produces the best known result.

Graph model: Graph, G(V,E), where V is the set of vertices and E is the set of edges.

Domains:

Input Domain: D_i includes the following:

V, a set of vertices

E, a set of edges

L, a set of locations

Fa, a set of facilities

D, a set of distances

Fl*, multiple sets of flows

Output Domain: D_o, Sequence of facilities/locations in best known solution

Size of search space:

For QAP: n^n where n is the number of facilities/locations.

For mQAP: n^{fn} where n is the number of facilities/locations and f is the number of different flows coming from the facilities.

Size of solution space:

For QAP: $n!$ where n is the number of facilities/locations.

For mQAP: $(fn)!$ where n is the number of facilities/locations and f is the number of different flows coming from the facilities.

Input Conditions:

V, E, L, Fa, D, and Fl are all non-negative values

$|L| = |Fa|$

Output Conditions:

$\forall\, l \,\exists\, fa \;\ni\; (number\, of\, fa = 1) \wedge \forall\, fa \,\exists l \;\ni\; (number\, of\, l = 1)$

0/1 Formulation:

For the 0/1 formulation, it is best handled by having a facility = zero when it has not been assigned a location and a one when it has been assigned. Then, as the algorithm goes through and fills the locations with facilities, it will be able to tell which ones have already been taken and which ones are available to be used.

D.2 Algorithm Domain Selection in Symbolic Framework with Design Variation

D.2.0.1 Stochastic Search - MOMGA-II. This section discusses the mapping of the algorithm domain to symbolic notation for the MOMGA-II program.

Algorithm Domain Requirements Specification Form:

For this part of the specification, the general genetic algorithm requirement is specified. This is the high level specification of the algorithm.

Name:

Stochastic-Search Fast Messy Genetic Algorithm

Domains:

D_s is a set of solutions, referred to as a population

The population size n is the cardinality of D_s

Operations:

I(x): $x \in D_s \wedge x \in S$ where S is the set of solutions.

O(x, z): $x \in D_s \wedge z \in D_s \wedge z \in S$ where S is the set of solutions.

Algorithm Domain Requirements Design Specification Form:

For the design specification form, the genetic algorithm is defined in more detail. This specification includes more information about the operators and data types that are needed for the genetic algorithm [80].

Name:

Stochastic-Search-FMGA (D_s)

Domains:

D_i is a set of algorithm-internal solutions

D_s is a set satisfying solutions

Imports:

ADT set

list

integer

Initialization of Feasible Solutions $\rightarrow D_s$

$D_i = \emptyset$

Operations:

I(x): $x \in D_s \wedge x \in S$ where S is the set of satisfying solutions.

O(x, z): $x \in D_s \wedge z \in D_s \wedge z \in S$ where S is the set of satisfying solutions.

Next Solution Generator $\rightarrow x$ for $x \in D_s \wedge D_s \in D_i$

Recombination (Cut & Splice) $x \rightarrow y$ with cut & splice probability

Mutation $x \rightarrow y$ with mutation probability

Feasibility $(y) \rightarrow$ boolean [if true $\vee(y, D_i)$] "genotype"

Fitness/objective function mapping $f(x)$ of each $x \in D_i$ "phenotype"

Selection $D_i \rightarrow D_s$ using $f(x)$ as criteria, $x \in D_i$

Algorithm Domain Function Specification Form:

For the function specification form, the genetic algorithm is defined in a high level, pseudocode format. This specification includes more information about the function of the genetic algorithm [80].

Function:

Stochastic-Search-FMGA (D_s)

Initial Condition:

Generate feasible $D_{initial} \rightarrow D_s$

$D_i = \emptyset$

Probability of cut & splice p_c

Probability of mutation p_m

Body:

While (not time/generation termination) do ss-fmga loop:

next-state-solution/population D_s, $D_s \rightarrow D_i$; Do for each $x \in D_s$, size n

Recombination(x) = y with p_c

Mutation(x) = y with p_m

if Feasibility(y) then $\vee(y, D_i) \rightarrow D_i$

Fitness calculation $f(x)$ for each $x \in D_i$

Selection(D_i) $\rightarrow D_s$ based upon $f(x)$, $x \in D_i$

end ss-fmga while loop

Find optimal $z \in D_s$

END Function

D.3 Algorithm Domain Pseudo Code to Implementation in chosen Language

D.3.0.2 MOMGA-II Pseudocode. In this section, the MOMGA-II pseudocode is laid out as presented by the author of the code [133].

For n = 1 to o

Perform Probabilistically Complete Initialization

Evaluate Each Population Member's Fitness (w.r.t. k Templates)

// *Building Block Filtering Phase*

For i = 1 to Maximum Number of Building Block Filtering Generations

If (Building Block Filtering Required Based off of Input Schedule)

Then Perform Building Block Filtering

Else

Perform Tournament Threshold Selection

Endif

End Loop

Juxtapositional Phase

For i = 1 to Maximum Number of Juxtapositional Generations

Cut-and-Splice

Evaluate Each Population Member's Fitness (w.r.t. k Templates)

Perform Tournament Threshold Selection and Fitness Sharing

$P_{known}(t) = P_{current}(t) \cup P_{known}(t-1)$

End Loop

Update k Competitive Templates (Using Best Value Known in Each Objective)

End Loop

Appendix E. Test Suite

This appendix lists the test suite used in the experiments for this research.

Table E.1 **Test Suite used - Knowles and Corne**

Test Name	Instance Category	# of locations	# of flows
KC10-2fl-1uni	Uniform	10	2
KC10-2fl-2uni	Uniform	10	2
KC10-2fl-3uni	Uniform	10	2
KC20-2fl-1uni	Uniform	20	2
KC20-2fl-2uni	Uniform	20	2
KC20-2fl-3uni	Uniform	20	2
KC30-3fl-1uni	Uniform	30	3
KC30-3fl-2uni	Uniform	30	3
KC30-3fl-3uni	Uniform	30	3
KC10-2fl-1rl	Real-like	10	2
KC10-2fl-2rl	Real-like	10	2
KC10-2fl-3rl	Real-like	10	2
KC10-2fl-4rl	Real-like	10	2
KC10-2fl-5rl	Real-like	10	2
KC20-2fl-1rl	Real-like	20	2
KC20-2fl-2rl	Real-like	20	2
KC20-2fl-3rl	Real-like	20	2
KC20-2fl-4rl	Real-like	20	2
KC20-2fl-5rl	Real-like	20	2
KC30-3fl-1rl	Real-like	30	3
KC30-3fl-2rl	Real-like	30	3
KC30-3fl-3rl	Real-like	30	3

Appendix F. Baseline Result Figures

This appendix contains many of the figures generated to show the results of the initial MOMGA-II runs.

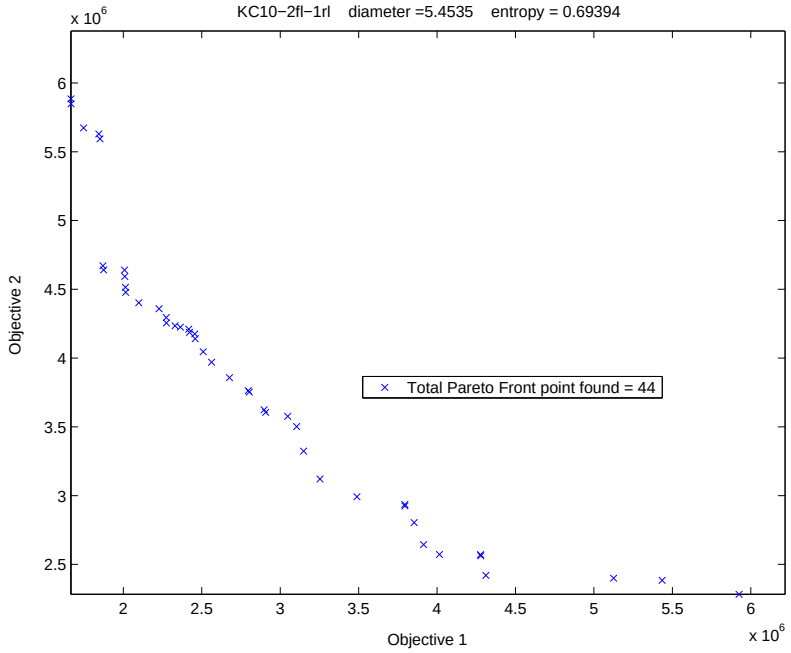

Figure F.1 Pareto front found for the KC10-2fl-1rl test instance using MOMGA-II initial settings

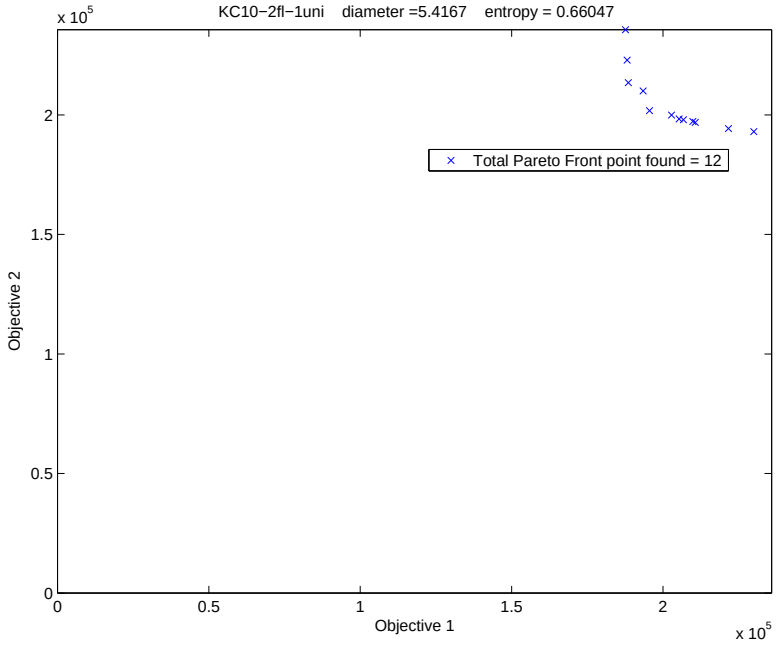

Figure F.2 Pareto front found for the KC10-2fl-1uni test instance using MOMGA-II initial settings

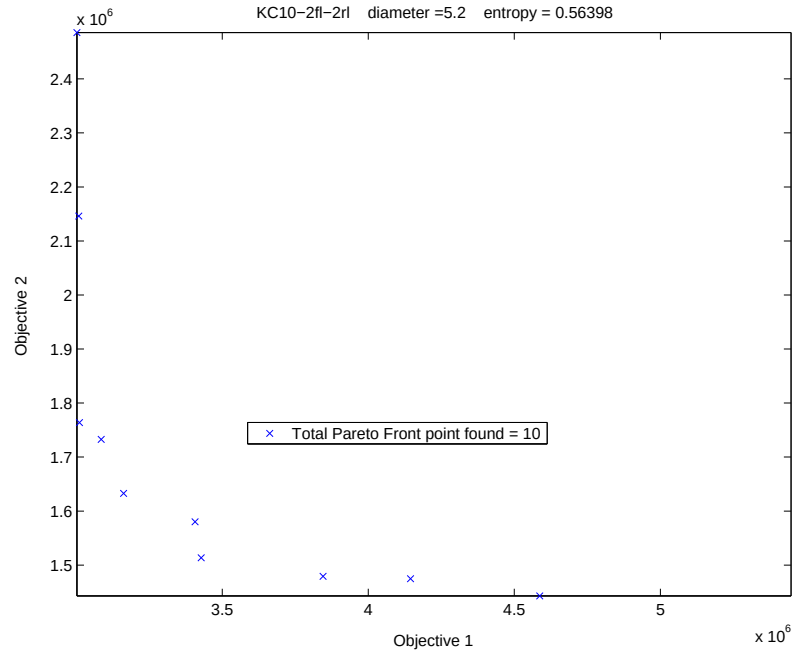

Figure F.3 Pareto front found for the KC10-2fl-2rl test instance using MOMGA-II initial settings

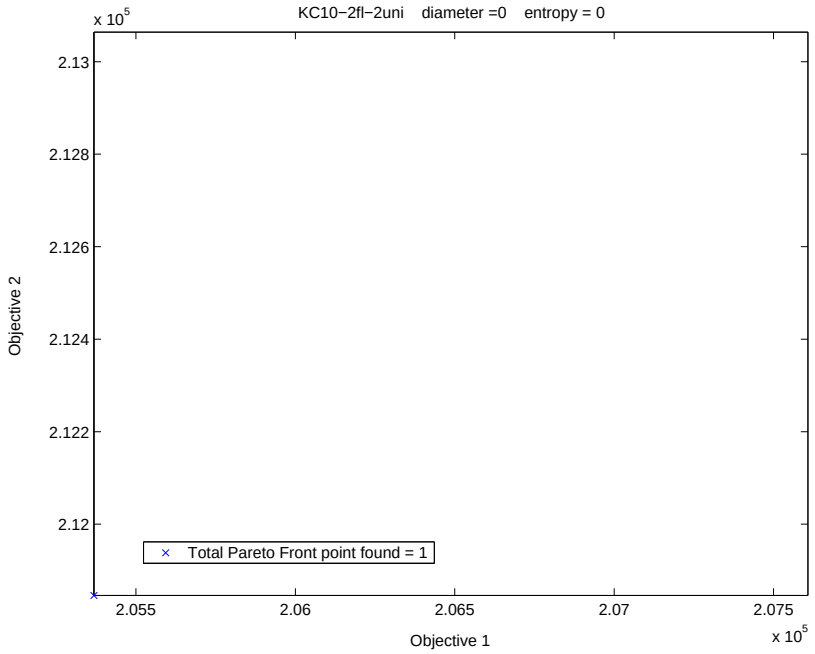

Figure F.4 Pareto front found for the KC10-2fl-2uni test instance using MOMGA-II initial settings

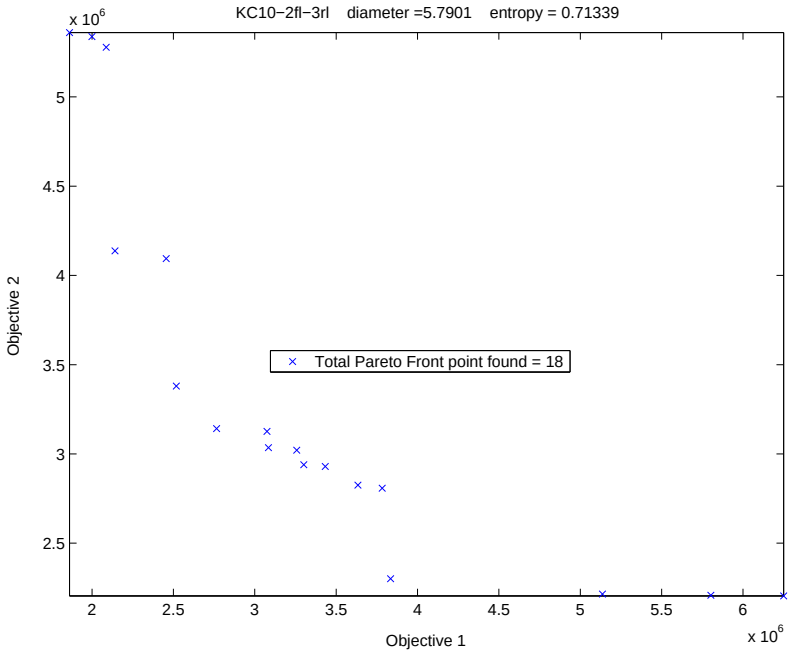

Figure F.5 Pareto front found for the KC10-2fl-3rl test instance using MOMGA-II initial settings

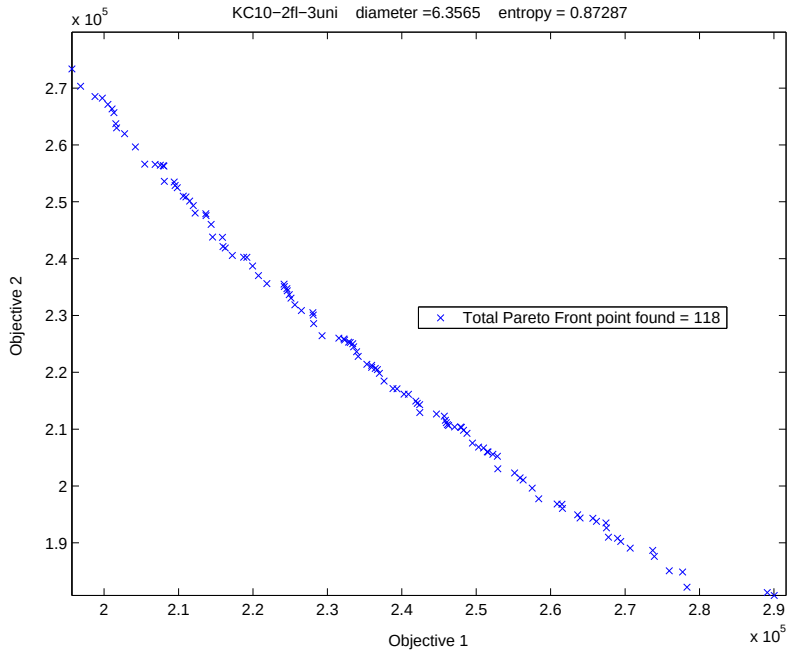

Figure F.6 Pareto front found for the KC10-2fl-3uni test instance using MOMGA-II initial settings

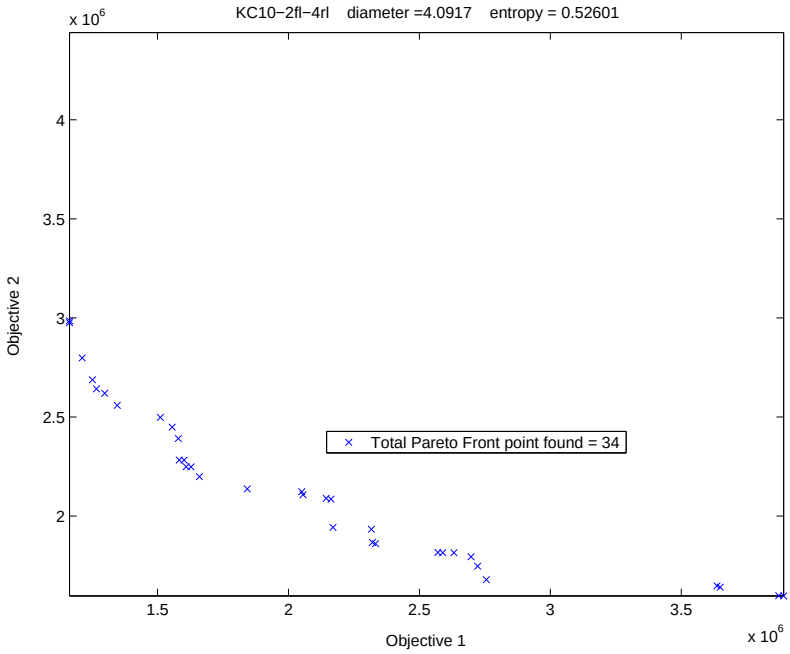

Figure F.7 Pareto front found for the KC10-2fl-4rl test instance using MOMGA-II initial settings

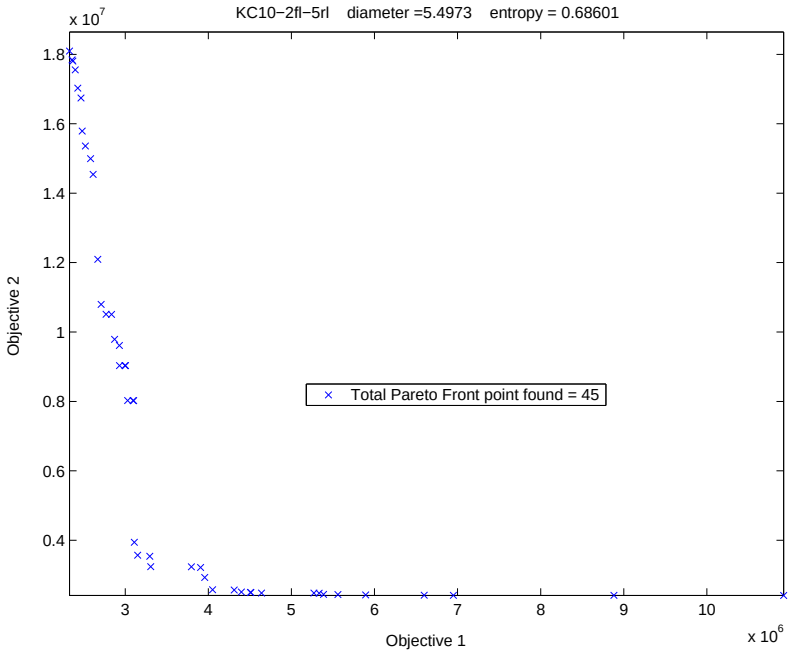

Figure F.8 Pareto front found for the KC10-2fl-5rl test instance using MOMGA-II initial settings

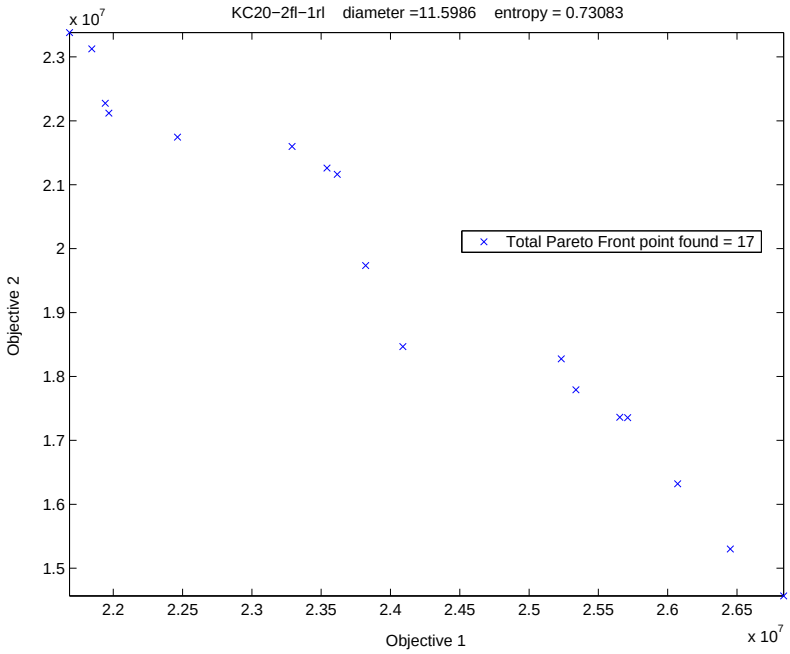

Figure F.9 Pareto front found for the KC20-2fl-1rl test instance using MOMGA-II initial settings

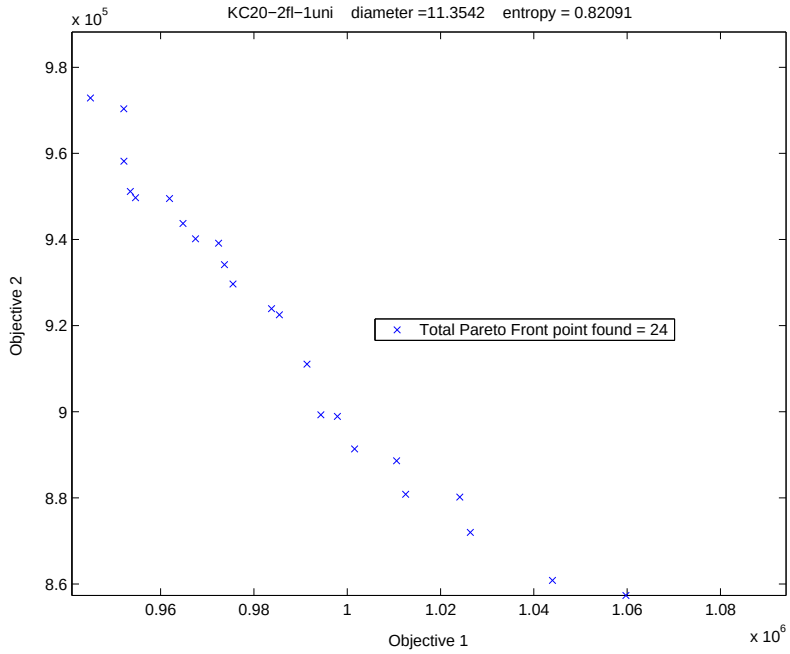

Figure F.10 Pareto front found for the KC20-2fl-1uni test instance using MOMGA-II initial settings

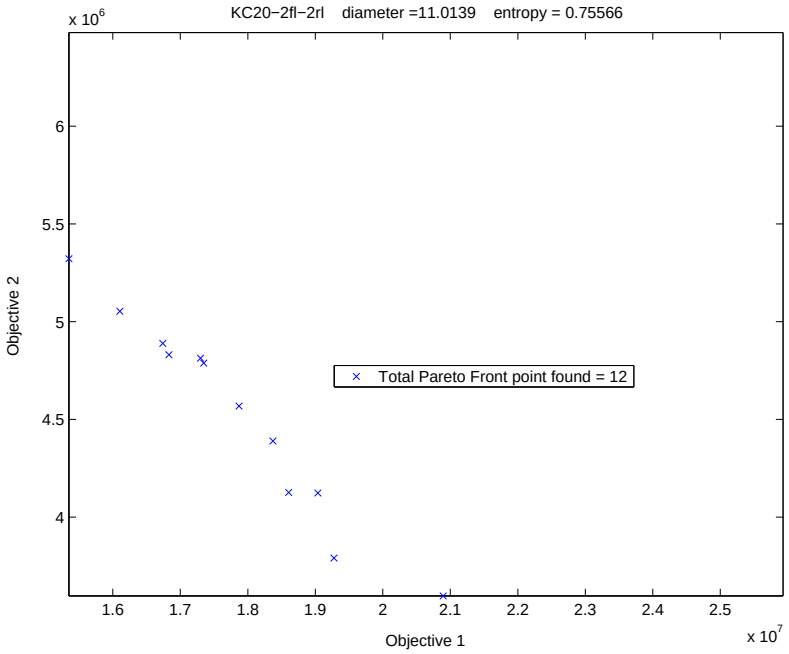

Figure F.11 Pareto front found for the KC20-2fl-2rl test instance using MOMGA-II initial settings

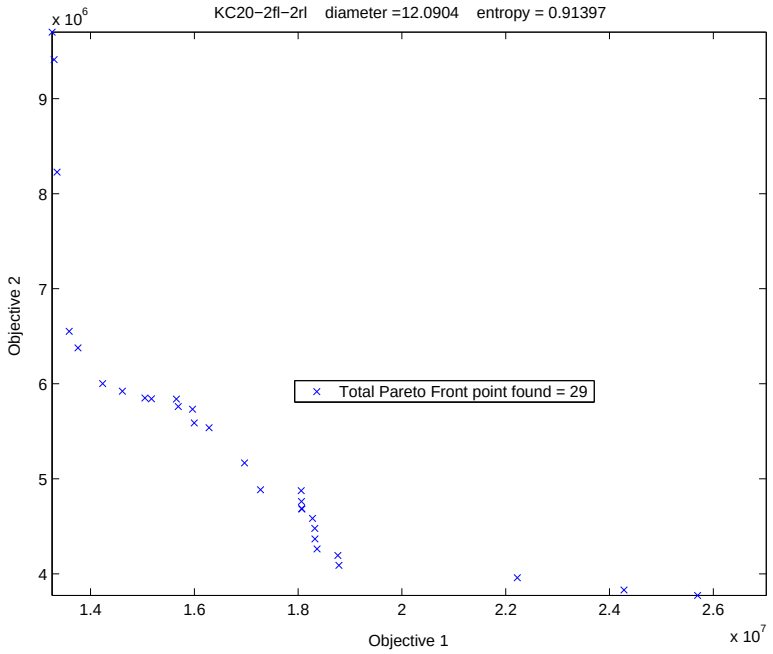

Figure F.12 Pareto front found for the KC20-2fl-3rl test instance using MOMGA-II initial settings

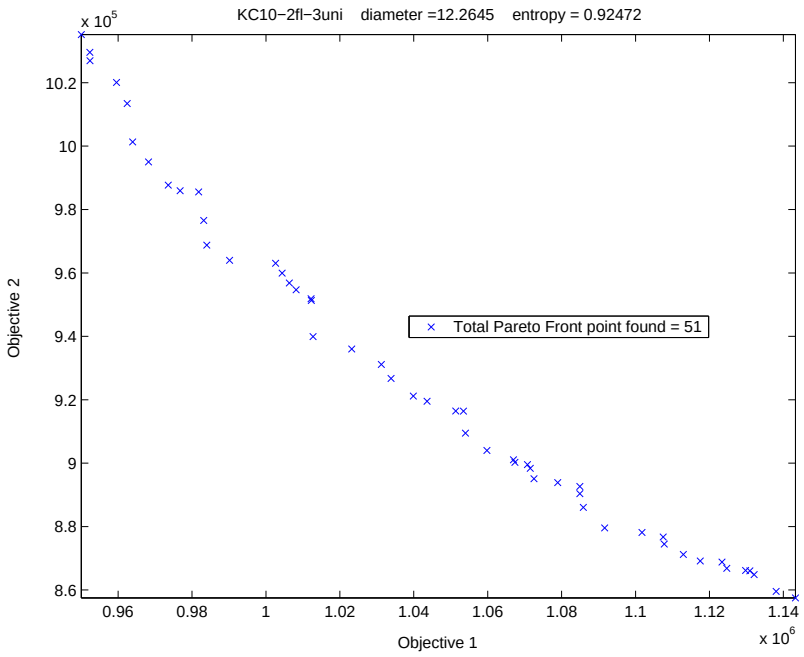

Figure F.13 Pareto front found for the KC20-2fl-3uni test instance using MOMGA-II initial settings

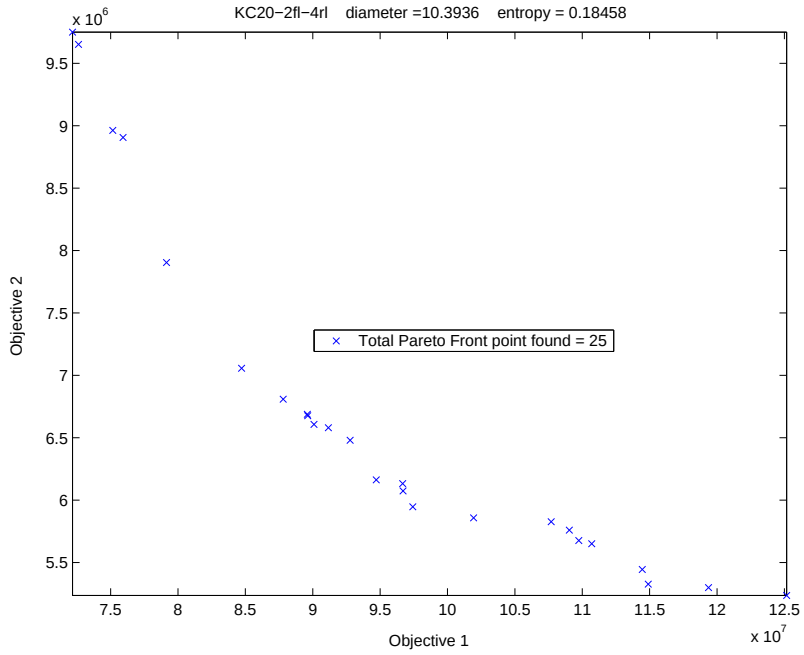

Figure F.14 Pareto front found for the KC20-2fl-4rl test instance using MOMGA-II initial settings

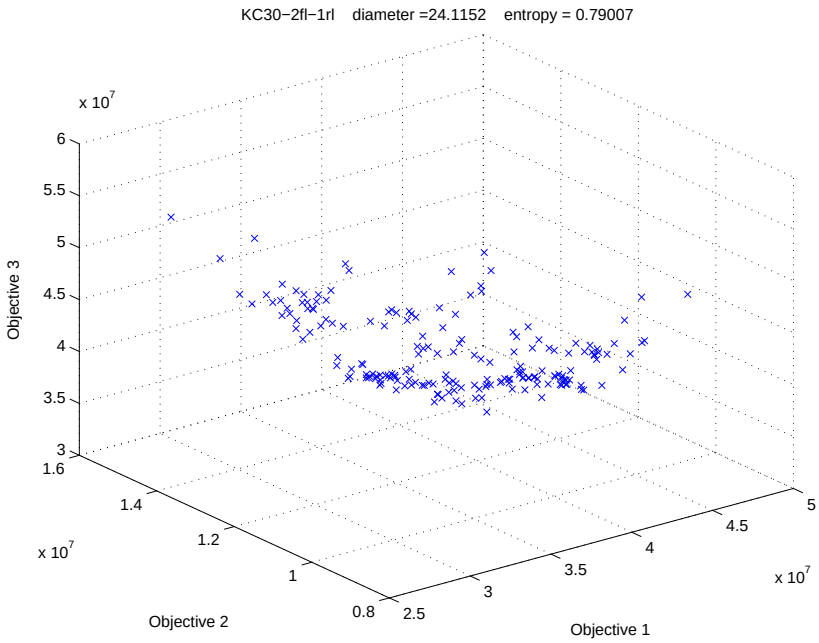

Figure F.15 Pareto front found for the KC30-3fl-1rl test instance using MOMGA-II initial settings

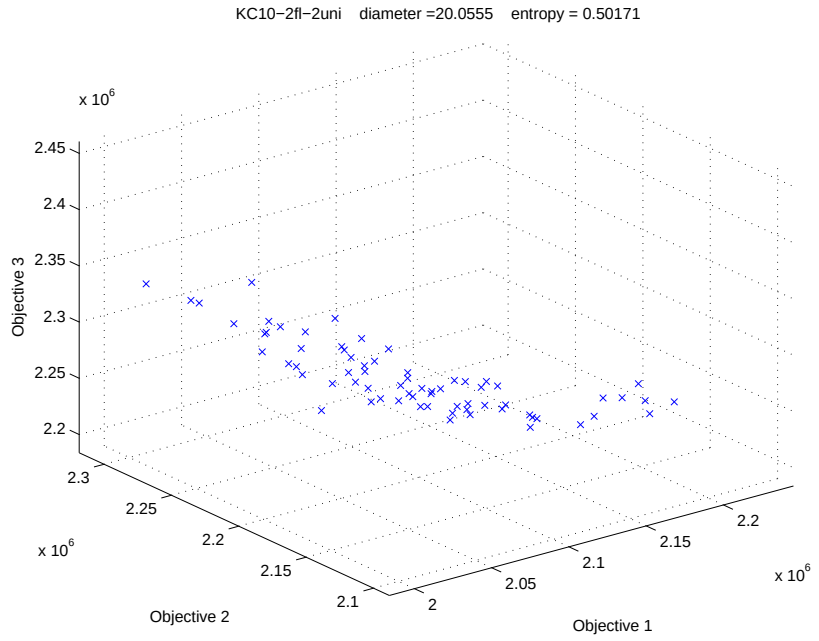

Figure F.16 Pareto front found for the KC30-3fl-1uni test instance using MOMGA-II initial settings

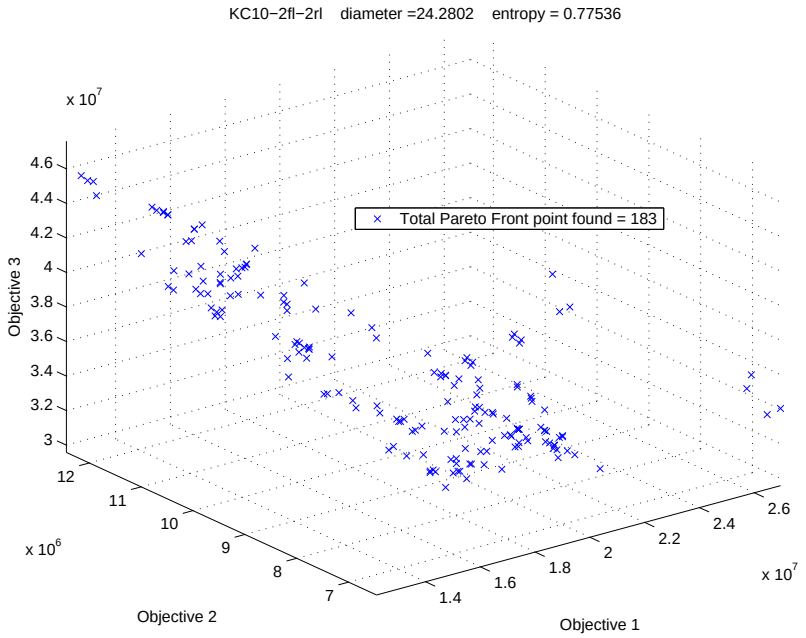

Figure F.17 Pareto front found for the KC30-3fl-2rl test instance using MOMGA-II initial settings

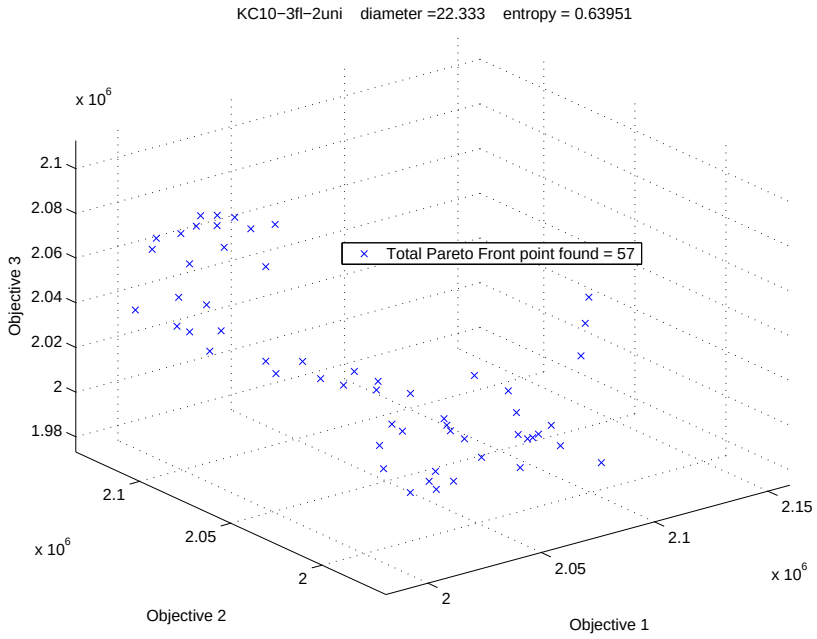

Figure F.18 Pareto front found for the KC30-3fl-2uni test instance using MOMGA-II initial settings

Appendix G. Graphs Comparing Two Methods of Running MOMGA-II

This appendix contains many of the figures generated to show the comparison of the results of the MOMGA-II using a randomized Competitive Template and a propagated Competitive Template.

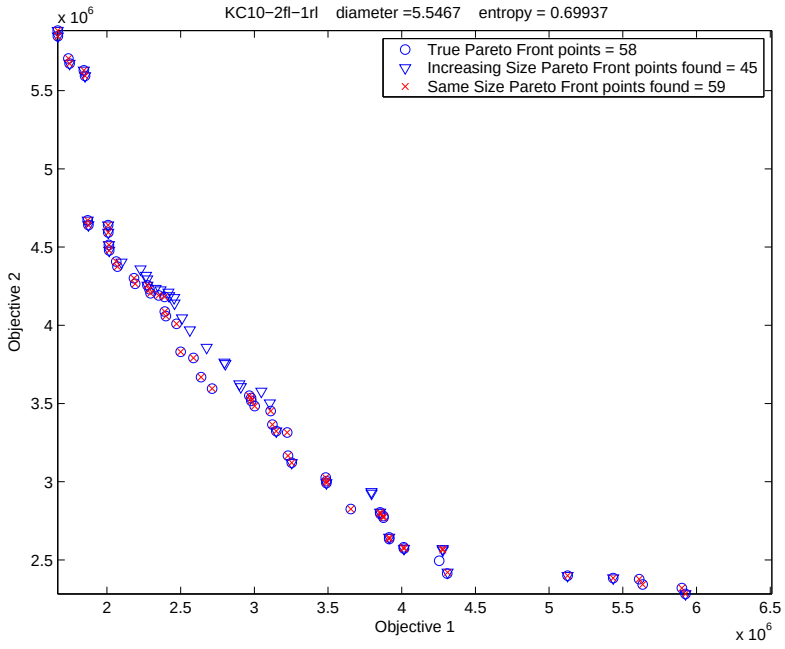

Figure G.1 Comparison of two MOMGA-II methods and PF_{true} found for the KC10-2fl-1rl test instance

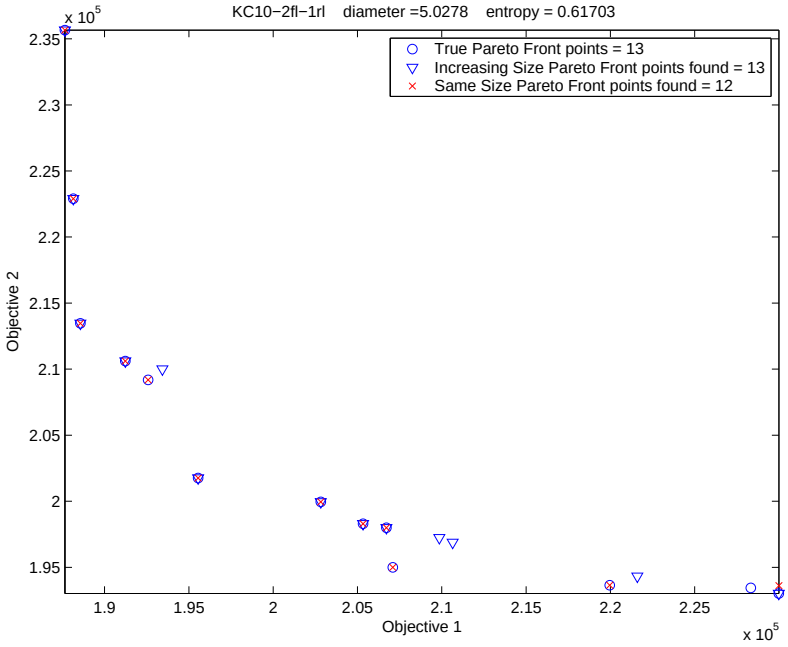

Figure G.2 Comparison of two MOMGA-II methods and PF_{true} found for the KC10-2fl-1uni test instance

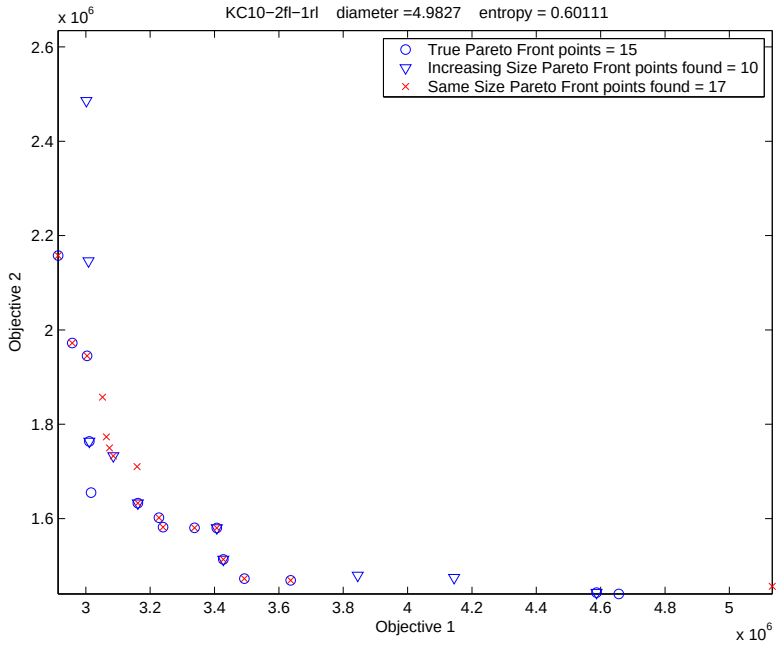

Figure G.3 Comparison of two MOMGA-II methods and PF_{true} found for the KC10-2fl-2rl test instance

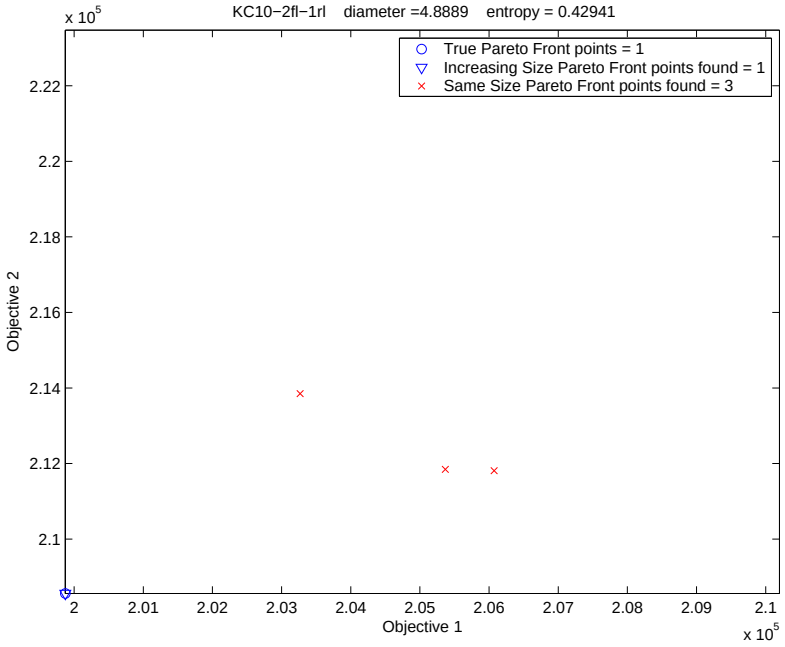

Figure G.4 Comparison of two MOMGA-II methods and PF_{true} found for the KC10-2fl-2uni test instance

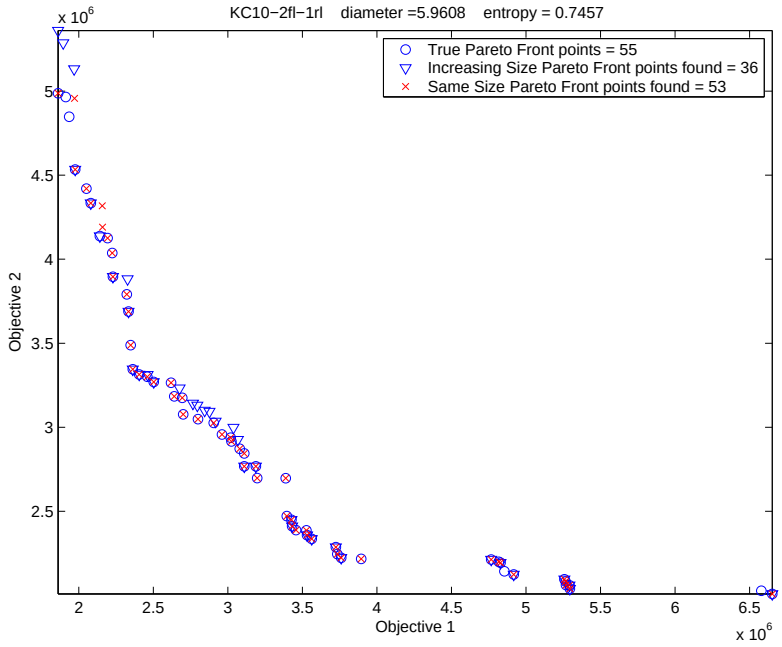

Figure G.5 Comparison of two MOMGA-II methods and PF_{true} found for the KC10-2fl-3rl test instance

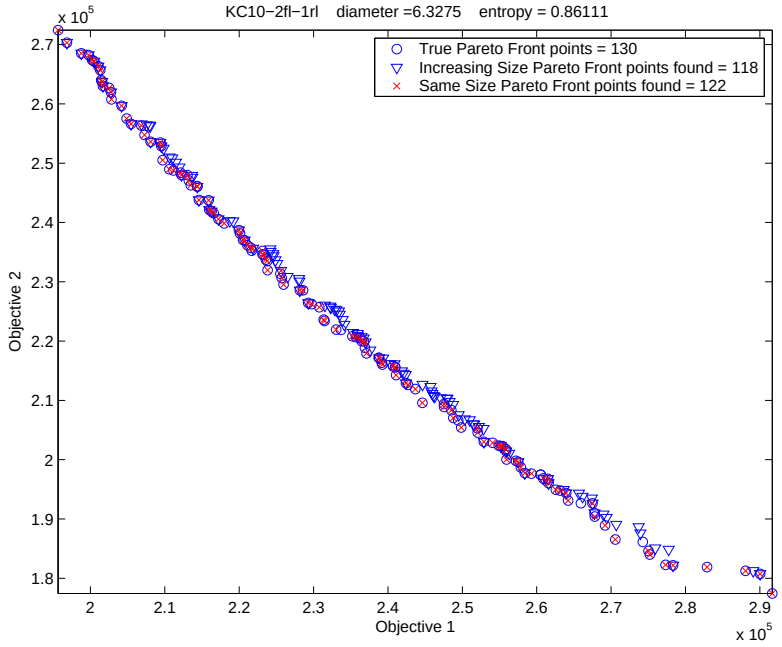

Figure G.6 Comparison of two MOMGA-II methods and PF_{true} found for the KC10-2fl-3uni test instance

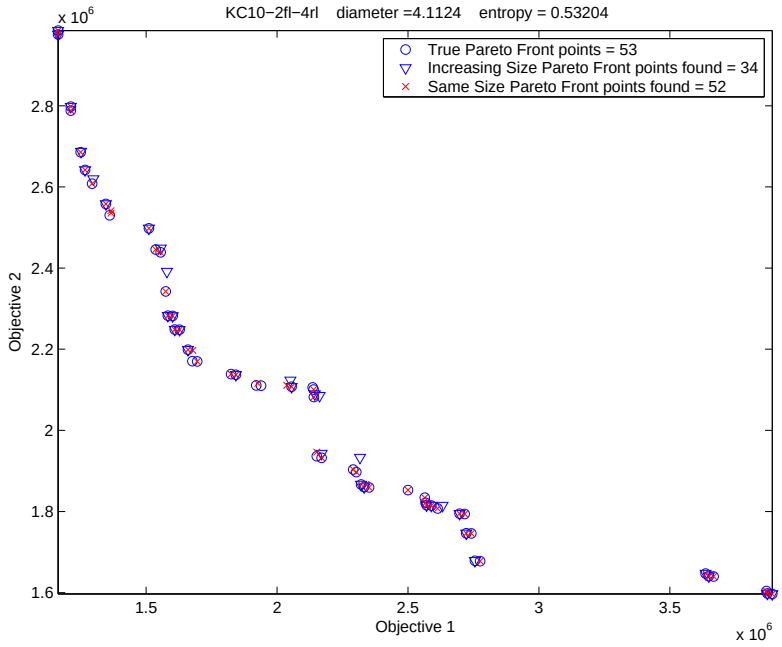

Figure G.7 Comparison of two MOMGA-II methods and PF_{true} found for the KC10-2fl-4rl test instance

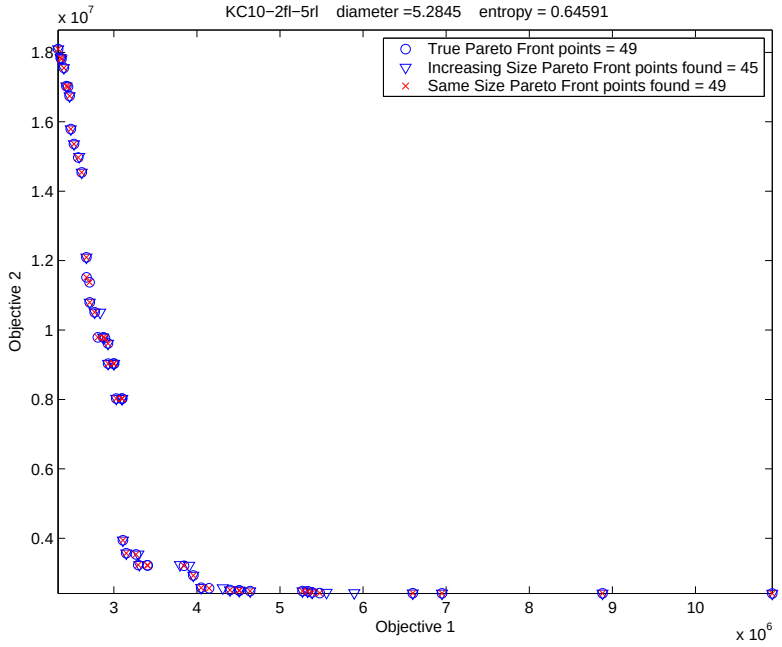

Figure G.8 Comparison of two MOMGA-II methods and PF_{true} found for the KC10-2fl-5rl test instance

Appendix H. Graphs Comparing Large and Small Building Block Sizes

This appendix contains many of the figures generated to illustrate how large and small building blocks populate the Pareto Front.

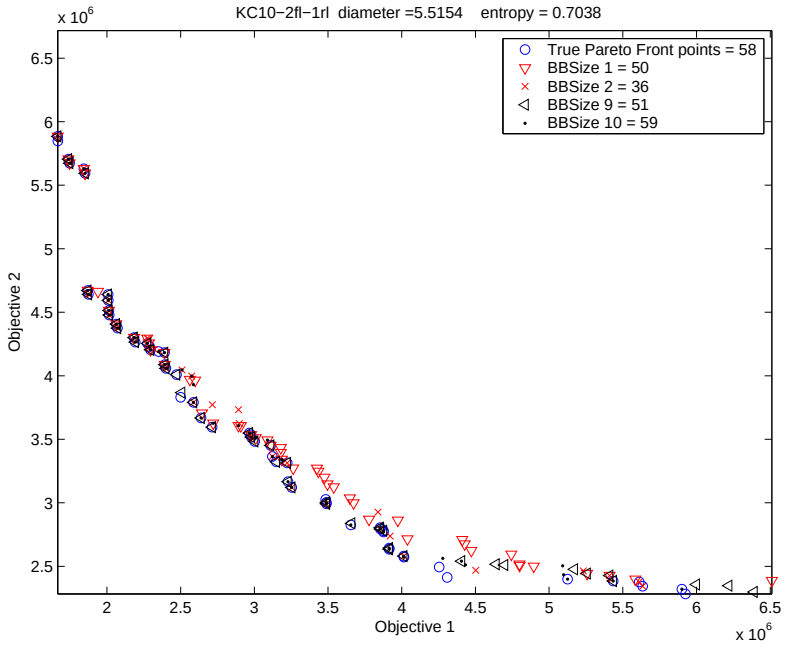

Figure H.1 Comparison of large and small building block sizes for the KC10-2fl-1rl test instance

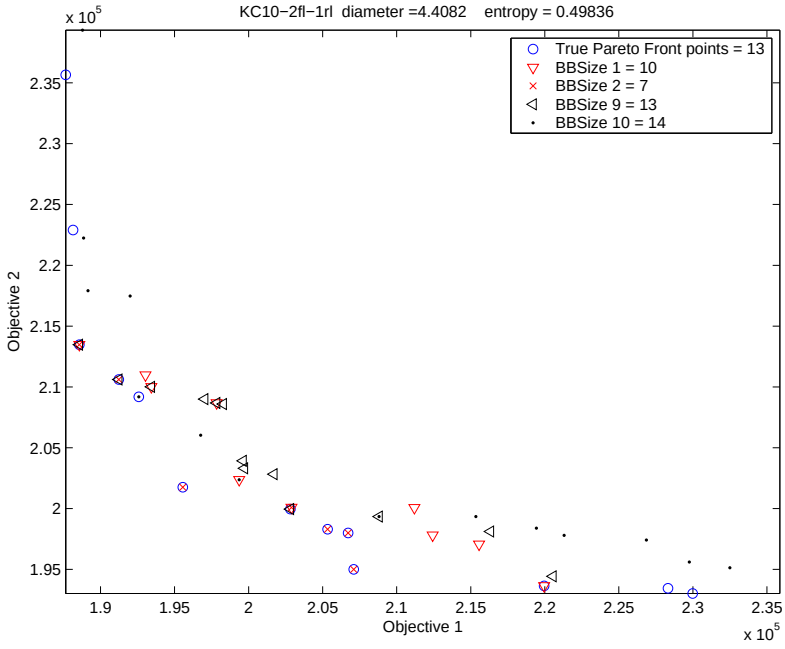

Figure H.2 Comparison of large and small building block sizes for the KC10-2fl-1uni test instance

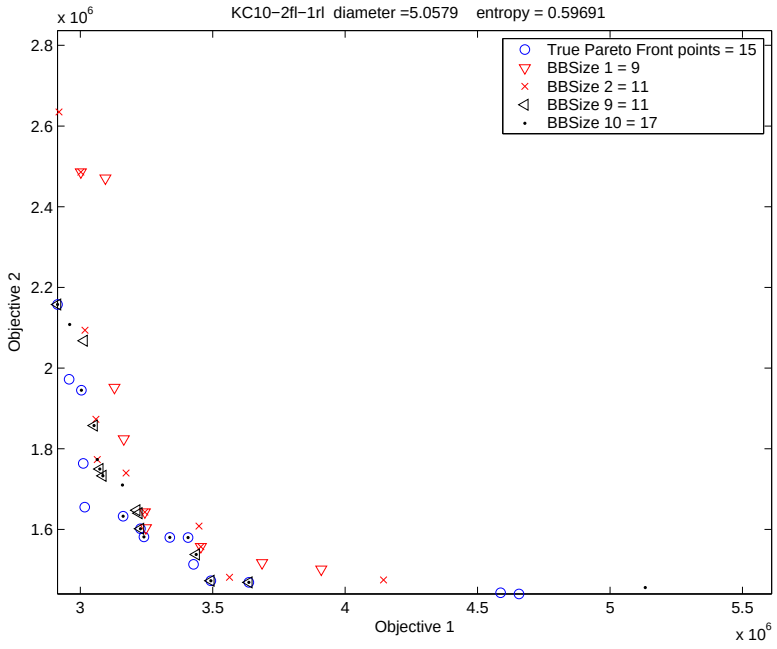

Figure H.3 Comparison of large and small building block sizes for the KC10-2fl-2rl test instance

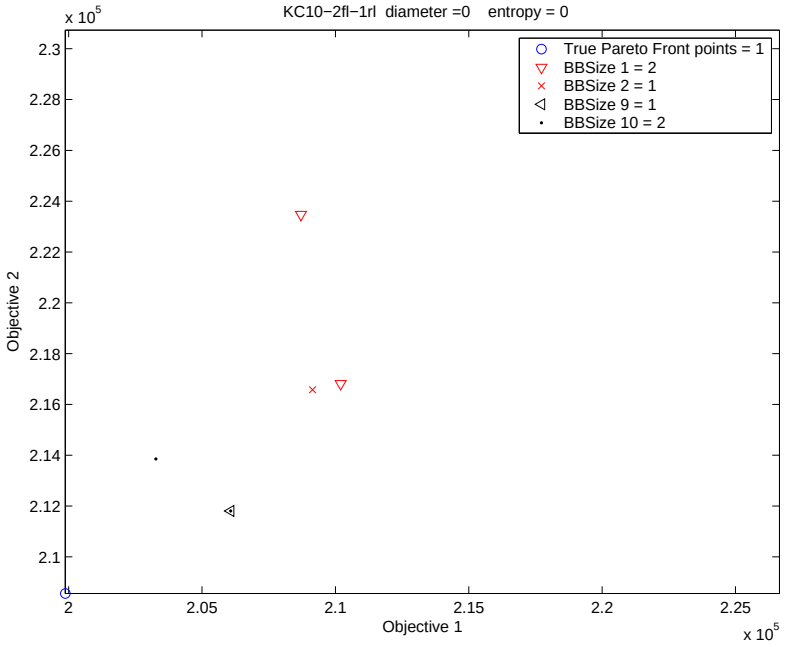

Figure H.4 Comparison of large and small building block sizes for the KC10-2fl-2uni test instance

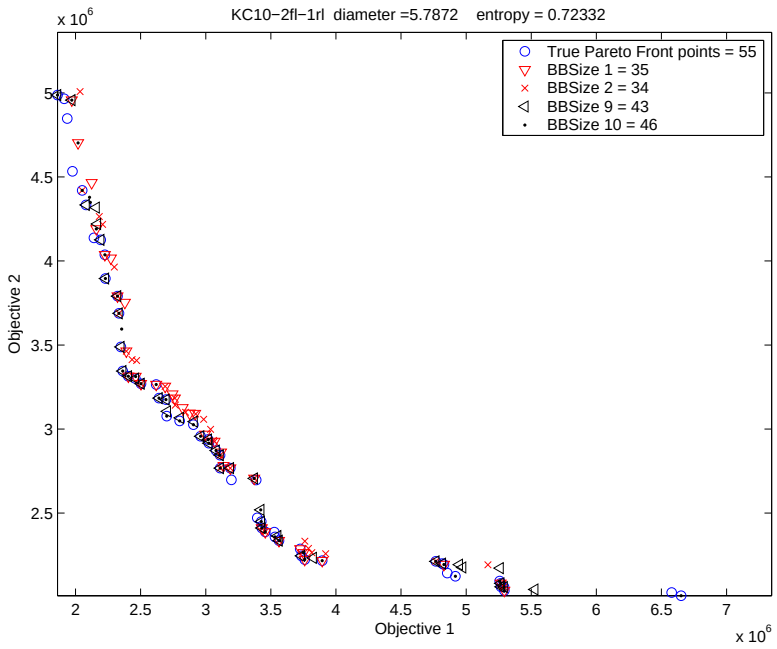

Figure H.5 Comparison of large and small building block sizes for the KC10-2fl-3rl test instance

Appendix I. Optimal Results

This appendix lists the optimal values found for all the 10 location and 10 facility instances in the test suite. All of the tables are sorted with respect to their first objective value. This is done so the reader can easily correlate the numbers to the graphs.

I.1 QAP1

Table I.1 lists the 58 True Pareto Front members found for the instance KC10-2fl-1rl. This instance has ten locations (formation locations) and ten facilities (UAVs). It also consists of two flows (communication channels). This "real-like" instance consists of input matrices with structured entries.

Table I.1: KC10-2fl-1rl Optimal Results

KC10-2fl-1rl Optimal Results											
Loc 1	Loc 2	Loc 3	Loc 4	Loc 5	Loc 6	Loc 7	Loc 8	Loc 9	Loc 10	Obj 1	Obj 2
6	2	4	5	1	7	3	9	8	10	1665490	5884156
6	1	4	5	2	7	3	9	8	10	1666856	5848128
4	2	6	5	1	7	3	9	8	10	1738338	5705362
4	1	6	5	2	7	3	9	8	10	1746720	5673390
4	2	6	5	1	7	3	8	9	10	1843432	5629660
4	1	6	5	2	7	3	8	9	10	1851614	5593954
6	2	7	4	1	3	9	10	8	5	1869616	4670952
6	1	7	4	2	3	9	10	8	5	1874454	4641012
6	2	4	3	1	7	8	10	9	5	2007808	4640018
6	1	4	3	2	7	8	10	9	5	2008988	4593588
6	2	4	7	1	3	9	10	8	5	2013520	4514312
6	1	4	7	2	3	9	10	8	5	2014886	4478284
4	2	6	7	1	3	9	10	8	5	2063048	4406976

Table I.1: *continued...*

KC10-2fl-1rl Optimal Results											
4	1	6	7	2	3	9	10	8	5	2071430	4375004
4	2	6	10	1	7	3	8	9	5	2183516	4301002
4	1	6	10	2	7	3	8	9	5	2191698	4265296
6	1	4	7	2	10	3	8	9	5	2274450	4254938
4	2	7	10	1	6	3	9	8	5	2288108	4229954
4	1	7	10	2	6	3	9	8	5	2295602	4204590
7	1	4	10	2	6	3	8	9	5	2350502	4190934
2	6	4	1	7	3	9	10	8	5	2390824	4181772
4	2	7	10	1	6	3	8	9	5	2391980	4087524
4	1	7	10	2	6	3	8	9	5	2399274	4058426
8	1	4	10	2	6	3	9	7	5	2471790	4008888
9	1	4	10	2	6	3	8	7	5	2500050	3830416
9	1	6	10	2	4	3	8	7	5	2587326	3791090
10	1	4	9	2	6	3	8	7	5	2638066	3668026
10	1	6	9	2	4	3	8	7	5	2713418	3594886
4	2	8	10	1	3	6	7	9	5	2965758	3549496
4	1	8	10	2	3	6	7	9	5	2977726	3534946
10	1	4	9	2	6	7	3	8	5	2979534	3516426
10	1	6	9	2	4	7	3	8	5	3001846	3483476
1	10	4	2	9	6	3	8	7	5	3111126	3451522
2	3	4	1	8	6	9	10	7	5	3122502	3364968
1	3	4	2	8	6	9	10	7	5	3148768	3322918
10	1	7	9	2	4	6	3	8	5	3222934	3314430
2	3	6	1	8	4	9	10	7	5	3228518	3166346
1	3	6	2	8	4	9	10	7	5	3253700	3121222
10	1	8	9	2	4	6	7	3	5	3484066	3026876
10	2	3	9	1	7	6	4	8	5	3489014	3003094

Table I.1: *continued...*

KC10-2fl-1rl Optimal Results											
10	1	3	9	2	7	6	4	8	5	3489050	2991410
10	1	3	9	2	4	6	7	8	5	3654466	2825750
8	4	10	9	6	7	1	2	3	5	3853632	2803788
8	4	10	9	6	7	2	1	3	5	3855644	2793660
9	4	10	8	6	7	1	2	3	5	3875598	2780242
9	4	10	8	6	7	2	1	3	5	3876464	2770116
10	4	8	9	6	7	1	2	3	5	3914472	2643796
10	4	8	9	6	7	2	1	3	5	3915342	2633674
10	7	8	9	6	4	1	2	3	5	4013702	2580854
10	7	8	9	6	4	2	1	3	5	4015088	2572208
10	4	3	9	6	7	2	1	8	5	4254472	2494600
10	7	3	9	6	4	2	1	8	5	4308066	2412590
1	3	7	2	8	5	9	10	6	4	5125520	2399260
1	4	6	2	10	5	9	8	7	3	5434522	2383876
2	3	7	1	10	5	9	8	6	4	5611252	2376806
1	3	7	2	10	5	9	8	6	4	5634982	2342680
2	4	7	1	10	5	9	8	6	3	5900254	2320190
1	4	7	2	10	5	9	8	6	3	5925064	2282788

I.2 QAP2

Table I.2 lists the 13 True Pareto Front members found for the instance KC10-2fl-1uni. This instance has ten locations (formation locations) and ten facilities (UAVs). It also consists of two flows (communication channels). This "uniformly random" instance consists of randomly generated input matrices.

Table I.2: KC10-2fl-1uni Optimal Results

KC10-2fl-1uni Optimal Results											
Loc 1	Loc 2	Loc 3	Loc 4	Loc 5	Loc 6	Loc 7	Loc 8	Loc 9	Loc 10	Obj 1	Obj 2
3	8	0	9	2	4	1	5	6	7	187646	235662
4	0	8	6	7	3	5	1	9	2	188150	222904
2	0	8	5	3	7	6	9	1	4	188568	213468
2	3	8	5	0	9	6	7	1	4	191232	210608
2	3	5	8	9	0	4	1	6	7	192584	209184
2	3	5	6	9	0	8	1	4	7	195552	201754
2	0	5	6	7	3	8	1	4	9	202824	199956
8	0	5	6	3	7	2	9	4	1	205332	198298
8	0	5	4	3	7	6	9	2	1	206716	197990
8	0	5	4	3	7	2	9	6	1	207094	195000
6	0	8	4	3	7	5	9	2	1	219958	193646
1	2	7	9	6	5	0	4	3	8	228322	193446
7	4	5	0	6	8	2	3	1	9	229986	193024

I.3 QAP3

Table I.3 lists the 15 True Pareto Front members found for the instance KC10-2fl-2rl. This instance has ten locations (formation locations) and ten facilities (UAVs). It also consists of two flows (communication channels). This "real-like" instance consists of input matrices with structured entries.

Table I.3: KC10-2fl-2rl Optimal Results

KC10-2fl-2rl Optimal Results											
Loc 1	Loc 2	Loc 3	Loc 4	Loc 5	Loc 6	Loc 7	Loc 8	Loc 9	Loc 10	Obj 1	Obj 2
1	6	4	2	10	9	3	7	5	8	2913670	2157762
3	10	4	7	8	6	5	9	2	1	2957598	1972302
3	7	2	4	8	9	5	6	10	1	3003812	1945206
2	4	10	1	7	6	5	8	3	9	3011006	1763676
2	4	10	1	9	6	5	8	3	7	3016380	1655122
3	7	2	10	6	9	5	4	1	8	3161774	1632768
7	4	10	1	9	6	5	8	3	2	3227068	1601818
8	4	10	2	7	6	1	9	5	3	3239972	1581630
7	4	10	3	9	6	1	8	5	2	3337864	1580278
8	4	10	2	6	9	1	7	5	3	3406874	1580046
8	4	10	2	9	6	1	7	5	3	3427484	1513460
8	7	2	10	6	9	1	4	5	3	3492962	1472730
8	7	2	4	6	9	1	10	5	3	3636620	1468960
6	2	5	10	9	7	1	4	8	3	4587008	1443068
4	2	5	10	9	7	1	6	8	3	4656190	1440264

I.4 QAP4

Table I.4 lists the one True Pareto Front members found for the instance KC10-2fl-2uni. This instance has ten locations (formation locations) and ten facilities (UAVs). It also consists of two flows (communication channels). This "uniformly random" instance consists of randomly generated input matrices.

Table I.4: KC10-2fl-2uni Optimal Results

KC10-2fl-2uni Optimal Results											
Loc 1	Loc 2	Loc 3	Loc 4	Loc 5	Loc 6	Loc 7	Loc 8	Loc 9	Loc 10	Obj 1	Obj 2
3	1	7	4	6	5	2	8	10	9	199872	208562

I.5 QAP5

Table I.5 lists the 55 True Pareto Front members found for the instance KC10-2fl-3rl. This instance has ten locations (formation locations) and ten facilities (UAVs). It also consists of two flows (communication channels). This "real-like" instance consists of input matrices with structured entries.

Table I.5: KC10-2fl-3rl Optimal Results

KC10-2fl-3rl Optimal Results											
Loc 1	Loc 2	Loc 3	Loc 4	Loc 5	Loc 6	Loc 7	Loc 8	Loc 9	Loc 10	Obj 1	Obj 2
3	6	9	7	5	1	8	10	2	4	1861090	4987620
1	2	10	4	5	3	8	9	6	7	1912342	4965330
3	6	9	7	2	1	8	10	5	4	1935968	4847916
3	6	9	7	5	1	4	10	2	8	1976046	4533340
3	6	9	7	2	1	4	10	5	8	2050904	4419672
3	10	9	7	5	1	4	6	2	8	2080766	4333574
3	4	9	7	5	1	6	10	2	8	2141324	4137342
3	10	9	7	5	1	6	4	2	8	2193494	4125534
3	4	9	7	2	1	6	10	5	8	2224832	4036688
3	6	7	9	5	1	4	10	2	8	2228838	3895882

Table I.5: *continued...*

KC10-2fl-3rl Optimal Results											
3	6	7	9	2	1	4	10	5	8	2322304	3790678
3	10	7	9	5	1	4	6	2	8	2333598	3688222
1	4	2	9	10	7	5	8	6	3	2347974	3488402
1	4	9	2	10	7	5	8	6	3	2362542	3345178
3	4	9	2	10	7	5	8	6	1	2405640	3313804
1	10	9	2	4	7	5	8	6	3	2459958	3300406
3	10	9	2	4	7	5	8	6	1	2502954	3269422
3	7	4	10	9	1	5	6	2	8	2618988	3264690
1	6	4	10	9	7	5	8	2	3	2640708	3183608
3	5	2	9	4	7	6	8	10	1	2694504	3175232
3	5	9	2	4	7	6	8	10	1	2700618	3077120
3	9	2	5	4	7	6	8	10	1	2799480	3048276
3	2	9	5	4	7	6	8	10	1	2905616	3026346
1	2	6	10	7	4	5	8	9	3	2960312	2957294
1	9	4	10	6	7	2	8	5	3	3018452	2938190
1	5	6	10	7	4	2	8	9	3	3024406	2915598
1	2	10	6	7	4	5	8	9	3	3079520	2872532
7	5	3	8	10	1	6	2	4	9	3110142	2844300
7	5	3	8	4	1	6	2	10	9	3110508	2767638
3	4	2	5	9	7	10	8	6	1	3188326	2767330
9	5	3	8	4	1	6	2	10	7	3196556	2697522
9	5	1	8	4	3	6	2	10	7	3387908	2697270
4	7	1	8	2	3	9	6	5	10	3395488	2471486
10	7	1	8	2	3	9	6	5	4	3426534	2449164
4	7	1	8	9	3	5	6	2	10	3430910	2411174
10	7	1	8	9	3	5	6	2	4	3456724	2388660
4	7	1	8	9	3	2	6	5	10	3526598	2386348

Table I.5: *continued...*

KC10-2fl-3rl Optimal Results											
4	7	1	8	5	3	9	6	2	10	3531472	2358148
10	7	1	8	5	3	9	6	2	4	3562518	2335826
4	2	1	8	7	3	5	6	9	10	3724928	2286944
4	9	1	8	7	3	5	6	2	10	3734182	2245668
10	9	1	8	7	3	5	6	2	4	3759996	2223154
10	9	1	8	7	3	2	6	5	4	3894492	2216840
7	4	3	5	8	1	10	2	6	9	4766708	2212576
9	4	1	5	8	3	10	2	6	7	4817046	2199526
7	6	3	5	8	1	10	2	4	9	4828998	2194122
9	4	3	5	8	1	10	2	6	7	4855160	2143022
9	6	3	5	8	1	10	2	4	7	4917628	2124232
4	5	7	8	1	3	2	6	9	10	5257502	2095212
4	9	7	8	1	3	5	6	2	10	5265382	2081946
4	9	7	8	1	3	2	6	5	10	5267876	2061436
10	9	7	8	1	3	5	6	2	4	5291194	2059458
10	9	7	8	1	3	2	6	5	4	5293688	2038948
9	4	3	5	1	8	10	2	6	7	6578114	2027476
9	6	3	5	1	8	10	2	4	7	6651336	2008658

I.6 QAP6

Table I.6 lists the 130 True Pareto Front members found for the instance KC10-2fl-3uni. This instance has ten locations (formation locations) and ten facilities (UAVs). It also consists of two flows (communication channels). This "uniformly random" instance consists of randomly generated input matrices.

Table I.6: KC10-2fl-3uni Optimal Results

KC10-2fl-3uni Optimal Results											
Loc 1	Loc 2	Loc 3	Loc 4	Loc 5	Loc 6	Loc 7	Loc 8	Loc 9	Loc 10	Obj 1	Obj 2
2	10	1	8	7	9	4	6	5	3	195648	272444
2	10	1	8	9	6	5	7	4	3	196836	270330
2	10	1	8	7	6	5	9	4	3	198774	268520
10	2	1	8	4	6	5	7	9	3	199760	268230
2	10	4	8	1	6	5	7	9	3	200226	267400
2	3	1	8	7	9	4	6	5	10	200448	267222
2	10	1	8	4	9	5	6	7	3	200522	267142
2	10	1	8	4	6	5	7	9	3	201066	266338
2	10	1	8	7	4	5	9	6	3	201320	265690
1	10	6	8	4	3	5	9	7	2	201446	263994
2	3	1	8	9	6	5	7	4	10	201574	263732
2	10	1	8	4	6	5	9	7	3	201686	263010
2	10	9	8	1	6	5	7	4	3	202528	262676
6	5	3	7	4	10	2	1	8	9	202754	261974
2	10	4	8	1	6	5	9	7	3	202826	260772
2	10	1	8	4	9	7	6	5	3	204196	259658
2	10	6	8	1	4	5	9	7	3	204868	257528
10	9	2	8	4	6	5	7	1	3	205460	256622
2	3	1	8	4	6	5	9	7	10	206842	256338
2	9	10	8	4	6	5	7	1	3	207276	254754
3	9	2	8	4	6	5	7	1	10	208094	253606
2	10	8	1	9	5	6	4	7	3	209412	253482
6	5	3	7	1	10	2	4	8	9	209538	252872
3	9	7	8	4	10	1	6	5	2	209714	250508

Table I.6: *continued...*

KC10-2fl-3uni Optimal Results											
3	9	6	8	4	10	1	7	5	2	210608	248988
3	9	6	8	4	2	1	7	5	10	211160	248788
1	10	8	2	7	4	5	9	6	3	212116	248398
2	3	6	8	4	1	5	9	7	10	212238	248006
10	3	6	8	1	4	5	9	7	2	213038	247990
6	5	8	7	1	4	2	3	10	9	213232	247104
6	9	8	7	4	1	10	3	2	5	213486	246290
9	6	4	7	8	1	10	3	2	5	214352	246120
7	9	4	6	8	1	10	3	2	5	214378	246030
6	9	4	7	8	1	10	3	2	5	214578	243784
2	10	8	1	4	5	6	9	7	3	215902	243740
9	5	6	4	7	2	10	1	8	3	215982	242134
1	3	8	2	7	4	5	9	6	10	216282	241874
2	3	8	1	4	5	6	9	7	10	216578	241562
9	5	7	4	6	2	10	1	8	3	217230	240562
7	9	8	6	4	1	10	3	2	5	217976	239856
8	3	1	10	7	4	5	9	6	2	219958	238702
9	10	2	4	8	5	6	1	7	3	220110	238142
8	3	4	2	6	1	5	9	7	10	220514	237046
1	3	8	10	7	4	5	9	6	2	220738	237008
7	9	4	5	8	1	10	3	2	6	221110	236156
9	5	1	7	4	8	10	3	2	6	221558	235712
8	3	4	2	7	1	5	9	6	10	221686	235236
8	3	4	2	9	1	5	7	6	10	223122	235190
9	2	8	4	1	5	6	10	7	3	223132	234628
7	5	9	4	6	2	10	1	8	3	223266	234534
2	10	5	4	8	9	6	1	7	3	223582	233742

Table I.6: *continued...*

KC10-2fl-3uni Optimal Results											
3	10	6	8	9	4	5	1	7	2	223768	233500
9	5	8	7	4	1	10	3	2	6	223812	231972
2	10	5	4	8	9	7	1	6	3	225548	231346
6	9	4	5	8	1	10	3	2	7	225716	230644
5	9	8	7	4	1	10	3	2	6	225976	229540
1	5	9	4	7	2	10	6	8	3	228160	228566
9	10	8	2	7	4	5	1	6	3	228584	228564
9	5	8	6	4	1	10	3	2	7	229294	226430
2	10	9	4	1	5	6	8	7	3	229746	226230
4	3	8	2	7	1	5	9	6	10	230776	225706
5	9	8	6	4	1	10	3	2	7	231362	223614
9	5	8	7	3	1	10	4	2	6	231478	223392
9	5	1	7	3	8	10	4	2	6	233040	221938
5	9	8	7	3	1	10	4	2	6	233700	221888
9	10	2	8	4	7	6	1	5	3	235224	220800
3	6	9	4	8	1	10	7	2	5	235696	220674
6	9	4	5	8	1	10	7	2	3	236238	220532
3	6	4	5	8	1	10	7	2	9	236464	220010
9	3	1	10	7	4	5	8	6	2	236742	219982
8	5	9	7	3	1	10	4	2	6	236900	218874
9	5	8	6	3	1	10	4	2	7	237128	217922
7	3	4	5	8	1	10	9	2	6	238748	217194
3	10	2	4	8	6	5	1	7	9	238820	217136
9	5	1	6	3	8	10	4	2	7	239170	216348
5	9	8	6	3	1	10	4	2	7	239254	216034
8	5	9	6	3	1	10	4	2	7	240694	215772
10	2	7	9	8	4	5	1	6	3	240970	215596

Table I.6: *continued...*

KC10-2fl-3uni Optimal Results											
3	7	4	5	8	1	10	9	2	6	240972	215582
3	6	4	5	8	1	10	9	2	7	241106	214264
6	3	4	5	8	1	10	9	2	7	242418	212924
3	10	6	9	8	4	5	1	7	2	242702	212626
3	10	2	4	8	7	5	1	6	9	243676	211910
3	10	7	9	8	4	5	1	6	2	244626	209590
8	2	4	9	1	7	6	10	5	3	247530	209380
3	10	6	5	8	4	9	1	7	2	247556	208884
9	5	8	3	4	1	10	6	2	7	248522	208348
10	3	6	7	2	4	5	8	1	9	248780	207046
9	5	8	3	7	1	10	4	2	6	249460	206614
1	6	4	2	7	8	3	5	9	10	249838	205424
7	10	8	3	9	4	5	1	6	2	252008	205316
7	10	3	9	8	4	5	1	6	2	252076	204566
5	10	8	3	6	4	9	1	7	2	252900	203026
2	9	5	6	3	8	10	4	1	7	254068	202840
3	10	4	5	8	7	9	2	6	1	254894	202372
3	10	4	5	8	7	6	2	9	1	255162	202286
7	9	4	2	6	8	10	5	1	3	255430	202276
6	9	4	2	5	8	10	7	1	3	255798	201790
10	3	4	9	1	7	6	8	5	2	255878	201750
7	5	1	2	6	8	3	4	9	10	255886	201418
10	3	7	6	2	4	5	8	1	9	255934	200030
3	10	7	6	2	4	5	8	1	9	257192	199806
1	6	4	2	7	8	10	5	9	3	257512	199630
10	5	7	9	1	8	3	4	2	6	257882	198682
4	6	1	2	7	8	3	5	9	10	258396	197742

Table I.6: *continued...*

KC10-2fl-3uni Optimal Results											
9	5	1	10	7	8	3	4	2	6	259298	197656
10	3	7	6	2	1	5	8	4	9	260524	197462
4	6	8	10	5	1	2	7	3	9	260530	197448
4	6	1	2	7	8	10	5	9	3	260870	196836
5	9	8	2	6	1	10	4	7	3	261518	196774
9	5	1	2	7	8	10	4	6	3	261582	196048
1	6	4	9	10	8	3	5	2	7	262570	194924
5	9	8	3	7	1	2	4	10	6	263148	194790
6	3	4	2	7	8	10	5	1	9	263950	194370
9	5	8	10	7	1	2	4	3	6	264224	193118
4	6	1	10	5	8	2	7	3	9	265940	192660
7	3	4	2	9	8	10	5	1	6	267498	192630
5	9	8	3	6	1	2	4	10	7	267766	190996
4	6	1	10	7	8	2	5	3	9	267804	190408
5	9	8	10	7	1	2	4	3	6	269182	188932
9	5	1	10	7	8	2	4	3	6	270560	186534
4	6	1	2	9	8	10	5	3	7	274264	186130
5	9	8	10	6	1	2	4	3	7	275024	184598
9	5	1	10	6	8	2	4	3	7	275210	184008
4	6	1	10	9	8	2	5	3	7	277322	182262
4	7	1	10	9	8	2	5	3	6	278326	182186
6	4	1	10	9	8	2	5	3	7	282902	181888
4	1	6	10	9	8	2	5	3	7	288074	181254
6	4	3	2	9	8	1	5	10	7	290038	180748
4	1	7	3	9	8	2	5	10	6	291658	177448

I.7 QAP7

Table I.7 lists the 53 True Pareto Front members found for the instance KC10-2fl-4rl. This instance has ten locations (formation locations) and ten facilities (UAVs). It also consists of two flows (communication channels). This "real-like" instance consists of input matrices with structured entries.

Table I.7: KC10-2fl-4rl Optimal Results

KC10-2fl-4rl Optimal Results											
Loc 1	Loc 2	Loc 3	Loc 4	Loc 5	Loc 6	Loc 7	Loc 8	Loc 9	Loc 10	Obj 1	Obj 2
7	3	5	6	1	9	4	8	10	2	1163544	2985428
7	3	2	6	1	9	4	8	10	5	1163548	2975626
7	8	5	6	1	9	4	3	10	2	1212004	2797828
7	8	2	6	1	9	4	3	10	5	1212008	2788026
3	7	5	6	1	9	4	8	10	2	1249698	2685612
7	5	8	6	1	4	9	2	10	3	1266590	2641918
7	5	3	6	1	4	9	2	10	8	1293446	2607974
3	7	5	6	1	4	9	8	10	2	1345974	2558124
3	5	8	6	1	4	9	2	10	7	1360770	2529958
8	7	5	6	1	4	9	3	10	2	1510552	2497896
7	8	2	1	6	9	4	3	10	5	1536566	2445606
7	8	2	10	6	9	4	3	1	5	1555434	2438810
3	7	5	1	6	9	4	8	10	2	1574684	2342416
7	5	8	1	6	4	9	2	10	3	1582720	2282558
7	5	8	10	6	4	9	2	1	3	1601190	2281932
7	5	3	1	6	4	9	2	10	8	1609310	2248614
7	5	3	10	6	4	9	2	1	8	1627780	2247988
3	7	5	1	6	4	9	8	10	2	1659690	2198968
3	5	8	1	6	4	9	2	10	7	1676494	2170598

Table I.7: *continued...*

KC10-2fl-4rl Optimal Results											
3	5	8	10	6	4	9	2	1	7	1694908	2169972
8	7	5	1	6	4	9	3	10	2	1824074	2138764
8	7	5	10	6	4	9	3	1	2	1842584	2136910
8	5	3	1	6	4	9	2	10	7	1920376	2110854
8	5	3	10	6	4	9	2	1	7	1938790	2110228
4	7	3	10	6	2	9	5	1	8	2055214	2107578
9	2	3	1	6	4	8	5	10	7	2135854	2105738
4	8	2	1	6	5	9	3	10	7	2139458	2100530
4	8	5	1	6	2	9	3	10	7	2140372	2082328
5	7	3	1	6	2	9	4	10	8	2151396	1936234
5	7	3	10	6	2	9	4	1	8	2169696	1932274
3	2	8	1	6	4	9	7	10	5	2290372	1903536
3	5	8	1	6	4	9	7	10	2	2302386	1897212
8	2	3	1	6	4	9	7	10	5	2320502	1866500
8	5	3	1	6	4	9	7	10	2	2332516	1860176
8	5	3	10	6	4	9	7	1	2	2350926	1859290
9	2	3	1	6	4	8	7	10	5	2499664	1852696
7	2	8	1	6	4	9	3	10	5	2563938	1834194
7	2	3	1	6	4	9	8	10	5	2566828	1821718
7	5	3	1	6	4	9	8	10	2	2570670	1815758
7	5	3	10	6	4	9	8	1	2	2589134	1814742
3	2	7	1	6	8	4	9	10	5	2612988	1807272
8	5	9	1	6	4	3	7	10	2	2696748	1794676
8	5	9	10	6	4	3	7	1	2	2716230	1794094
3	5	9	1	6	4	8	7	10	2	2722318	1746396
3	5	9	10	6	4	8	7	1	2	2741800	1745814
3	5	9	1	6	8	4	7	10	2	2755568	1678694

Table I.7: *continued...*

KC10-2fl-4rl Optimal Results											
3	5	9	10	6	8	4	7	1	2	2775078	1677656
8	2	3	1	9	4	6	7	10	5	3636188	1647116
8	5	3	1	9	4	6	7	10	2	3648202	1640792
8	5	3	10	9	4	6	7	1	2	3666612	1639906
7	2	3	1	9	4	6	8	10	5	3868214	1603914
7	5	3	1	9	4	6	8	10	2	3872056	1597954
7	5	3	10	9	4	6	8	1	2	3890520	1596938

I.8 QAP8

Table I.8 lists the 49 True Pareto Front members found for the instance KC10-2fl-5rl. This instance has ten locations (formation locations) and ten facilities (UAVs). It also consists of two flows (communication channels). This "real-like" instance consists of input matrices with structured entries.

Table I.8: KC10-2fl-5rl Optimal Results

KC10-2fl-5rl Optimal Results											
Loc 1	Loc 2	Loc 3	Loc 4	Loc 5	Loc 6	Loc 7	Loc 8	Loc 9	Loc 10	Obj 1	Obj 2
10	3	1	7	2	5	4	9	6	8	2328942	18642252
10	3	1	7	2	5	6	9	4	8	2330402	18094754
10	3	1	7	2	5	9	4	6	8	2359984	17839446
10	7	1	3	2	5	6	9	4	8	2369348	17808138
10	7	1	3	2	5	9	4	6	8	2398982	17552796
10	3	1	7	6	5	2	9	4	8	2428796	17027890

Table I.8: *continued...*

KC10-2fl-5rl Optimal Results											
10	3	1	7	9	5	2	4	6	8	2449122	17012570
10	7	1	3	6	5	2	9	4	8	2466980	16741298
8	3	1	7	2	5	9	4	6	10	2482346	15789686
8	7	1	3	2	5	9	4	6	10	2521342	15358350
8	3	1	7	9	5	2	4	6	10	2571668	14970754
8	7	1	3	9	5	2	4	6	10	2613712	14539322
9	3	1	7	2	5	4	10	6	8	2669980	12094682
9	3	1	7	2	5	6	10	4	8	2671740	11517034
9	7	1	3	2	5	4	10	6	8	2708848	11374202
9	7	1	3	2	5	6	10	4	8	2710608	10796588
9	3	1	7	6	5	2	10	4	8	2769594	10509192
9	7	1	3	6	5	2	10	4	8	2807700	9788770
9	7	1	3	6	5	2	8	10	4	2872928	9786840
9	7	1	3	4	5	2	10	6	8	2893498	9769524
9	1	3	7	2	5	4	10	6	8	2930040	9610948
9	1	3	7	2	5	6	10	4	8	2931800	9033356
9	1	3	7	2	5	6	8	10	4	2997880	9031936
9	1	7	3	2	5	6	10	4	8	3004136	9030198
9	1	3	7	6	5	2	10	4	8	3029654	8025666
9	1	3	7	6	5	2	8	10	4	3094848	8023446
9	1	7	3	6	5	2	10	4	8	3101228	8022508
1	7	6	3	4	9	8	2	10	5	3109740	3943170
1	7	6	3	8	9	4	2	10	5	3149268	3567154
1	7	6	3	8	4	9	2	10	5	3266428	3534550
1	7	6	3	9	8	10	2	4	5	3293530	3231608
1	7	6	3	4	10	8	2	9	5	3402756	3220954
1	7	6	3	9	10	8	2	4	5	3407714	3215066

Table I.8: *continued...*

KC10-2fl-5rl Optimal Results											
1	7	6	3	4	10	8	9	2	5	3843816	3207366
3	7	6	1	8	9	4	2	10	5	3956508	2926878
3	7	9	1	4	8	10	2	6	5	4051608	2574304
3	7	9	1	4	10	8	2	6	5	4146592	2557762
3	7	5	1	9	8	10	2	4	6	4398266	2507500
3	7	5	1	4	10	8	2	9	6	4507746	2496996
3	7	5	1	9	10	8	2	4	6	4512450	2491132
7	3	5	1	2	10	8	4	6	9	4641292	2478828
7	3	5	1	4	10	8	2	6	9	5271006	2475908
7	3	5	1	2	10	8	9	6	4	5333146	2472814
7	3	2	5	4	8	10	9	6	1	5387394	2440766
7	3	2	5	4	10	8	9	6	1	5477662	2421724
3	7	4	9	2	10	8	5	6	1	6598854	2417386
7	3	4	9	2	10	8	5	6	1	6949694	2411920
7	3	9	4	2	10	8	5	6	1	8882256	2410430
7	3	9	4	5	10	8	1	6	2	10923682	2410292

Bibliography

1. Ahuja, R. K., Orlin, J. B., and Tiwari, A. "A Greedy Genetic Algorithm for the Quadratic Assignment Problem." *Computers and Operations Research27*. 917 – 934. 2000.

2. Air Force Research Laboratory / Information Directorate Public Affairs, , "AFRL/IF Mission Statement," January 2004. AFRL/IF, Wright Patterson AFB, OH, 2004, `http://www.rl.af.mil/mission/IF_miss.html`; accessed February 27, 2004.

3. Anstreicher, K., Brixius, N., Goux, J.-P., and Linderoth, J. "Solving Large Quadratic Assignment Problems on Computational Grids," *Mathematical Programming, Series B*, *91*:563 – 588 (2002).

4. Bachelet, V. *Métaheuristiques Parallèles Hybrides: Application au Problème D'affectation Quadratique*. PhD dissertation, Université des Sciences et Technologies de Lille, December 1999.

5. Bachelet, V. and Talbi, E.-G. "COSEARCH: a Co-evolutionary Metaheuristic." *Proceedings of the 2000 Congress on Evolutionary Computation*. 1550 – 1557. Piscataway, NJ: IEEE Service Center, 2000.

6. Bachelet, V. and Talbi, E.-G. "A Parallel Co-evolutionary Metaheuristic," *IPDPS Workshops 2000*, (628 - 635) (2000).

7. Bäck, T. A. *Evolutionary Algorithms in Theory and Practice*. New York - Oxford: Oxford University Press, 1996.

8. Benjaafar, S. "Design of Manufacturing Plant Layouts with Queueing Effects." *Proceddings of the 1998 IEEE International Conference on Robotics and Automation*. 260 – 265. May 1998.

9. Bosker, A. J., "UAVs role will expand, says General Jumper." Air Force Print News, November 2001, `http://www.af.mil/news/Nov2001/n20011005_1414.shtml`; accessed February 26, 2004.

10. Brüngger, A., Marzetta, A., Clausen, J., and Perregaard, M. "Joining Forces in Solving Large-Scale Quadratic Assignment Problems in Parallel." *Proceedings of the 11th International Parallel Processing Symposium*. 418 –427. 1997.

11. Burkard, R. E. "Selected topics on assignment problems," *Discrete Appl. Math.*, *123*(1-3):257–302 (2002).

12. Burkard, R. E. and Cela, E. "Linear Assignment Problem and Extensions," *Handbook of Combinatorial Optimization*, *4*:75 – 149 (1998). `http://citeseer.nj.nec.com/burkard98linear.html`; accessed January 17, 2004.

13. Burkard, R. E., Cela, E., Pardalos, P. M., and Pitsoulis, L. S. "The Quadratic Assignment Problem." *Handbook of Combinatorial Optimization 2*, edited by D.-Z. Du and P. M. Pardalos, 241–337, Kluwer Academic Publishers, 1998.

14. Burkard, R. E., Karisch, S. E., and Rendl, F. "QAPLIB - A Quadratic Assignment Problem Library," *European Journal of Operational Research*, *55*:115 – 119 (1991). `http://www.seas.upenn.edu/qaplib/`; accessed July 17, 2003.

15. Caswell, D. J. and Lamont, G. B. "Wire-Antenna Geometry Design with Multiobjective Genetic Algorithms." *Congress on Evolutionary Computation (CEC'2002)1*. 103–108. Piscataway, New Jersey: IEEE Service Center, May 2002.

16. Çela, E. *The Quadratic Assignment Problem - Theory and Algorithms*. Boston, MA: Kluwer Academic Publishers, 1998.

17. Coello, C. A. C., "List of References on Evolutionary Multiobjective Optimization." Laboratorio Nacional De Informatica Avanzada (LANIA), 2004, `http://www.lania.mx/~ccoello/EMOO/EMOObib.html`; accessed February 26, 2004.

18. Coello Coello, C. A. "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques," *Knowledge and Information Systems. An International Journal*, *1*(3):269–308 (August 1999).

19. Coello Coello, C. A. "A Short Tutorial on Evolutionary Multiobjective Optimization." *First International Conference on Evolutionary Multi-Criterion Optimization* edited by Eckart Zitzler, et al., 21–40, Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

20. Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B. *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Kluwer Academic Publishers, May 2002.

21. Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B. *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Kluwer Academic Publishers, May 2002. ISBN 0-3064-6762-3.

22. Cung, V.-D., Mautor, T., Michelon, P., and Tavares, A. "A Scatter Search Based Approach for the Quadratic Assignment Problem." *Proceedings of IEEE International Conference on Evolutionary Computation and Evolutionary Programming*. 165 – 170. 1997.

23. Das, I. and Dennis, J. "A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generattion in Multicriteria Optimization Problems." *Structural Optimization 14*, 63–69, 1997.

24. Day, R. O. *A Multiobjective Approach Applied to the Protein Structure Prediction Problem*. MS thesis, Air Force Institute of Technology, Wright Patterson AFB, OH, March 2002.

25. Day, R. O., Kleeman, M. P., and Lamont, G. B. "Solving the Multi-objective Quadratic Assignment Problem Using a fast messy Genetic Algorithm." *Congress on Evolutionary Computation (CEC'2003)4*. 2277–2283. Piscataway, New Jersey: IEEE Service Center, December 2003.

26. Day, R. O., Lamont, G. B., and Pachter, R. "Protein Structure Prediction by Applying an Evolutionary Algorithm." *HiCOMB 2003, Second IEEE International Workshop on High Performance Computational Biology*. 54–61. April 2003.

27. Day, R. O., Zydallis, J. B., and Lamont, G. B. "Solving the Protein Structure Prediction Problem through a Multiobjective Algorithm." *International Conference on Computational Nanoscience and Nanotechnology (ICCN)*. 32 – 35. April 2002.

28. Deb, K. "Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design." *Evolutionary Algorithms in Engineering and Computer Science* edited by Kaisa Miettinen, et al., chapter 8, 135–161, Chichester, UK: John Wiley & Sons, Ltd, 1999.

29. Deb, K. and Goldberg, D. E. *mGA in C: A Messy Genetic Algorithm in C*. Technical Report 91008, Illinios Genetic Algorithms Laboratory (IlliGAL), September 1991.

30. Defense Advanced Research Projects Agency, , "DARPA and Air Force Select Boeing to Build UCAV Demonstrator System." Defense Advanced Research Projects Agency, 1999, `http://www.defenselink.mil/news/Mar1999/b03241999\_bt123-99.html`; accessed June 10, 2003.

31. Dozier, G. V., McCullough, S., Homaifar, A., and Moore, L. "Multiobjective Evolutionary Path Planning via Fuzzy Tournament Selection." *IEEE International Conference on Evolutionary Computation (ICEC'98)*. 684–689. Piscataway, New Jersey: IEEE Press, May 1998.

32. Drezner, Z., Hahn, P. M., and Éric D. Taillard, . "A Study of Quadratic Assignment Problem Instances that are Difficult for Meta-heuristic Methods." Invited for submission to a special issue of *Annals of Operations Research*, devoted to the State-of-the-Art in Integer Programming, 2003.

33. Edwards Public Affairs, , "Global Hawk." Air Force Flight Test Center, 1998, `http://www.edwards.af.mil/articles98/docs_html/splash/may98/cover/ghawk%.htm`; accessed October 27, 2002.

34. Erickson, M., Mayer, A., and Horn, J. "The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems." *First International Conference on Evolutionary Multi-Criterion Optimization* edited by Eckart Zitzler, et al., 681–695, Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

35. Eshelman, L. J. "Genetic Algorithms." *Evolutionary Computation 1: Basic Algorithms and Operators* edited by T. Bäck, et al., chapter 8, 64–80, Bristol: Institute of Physics Publishing, 2000.

36. Falco, I. D., Balio, R. D., and Tarantino, E. "Testing Parallel Evolution Strategies on the Quadratic Assignment Problem," *Proceedings of the IEEE 1993 International Conference on Systems, Man, and Cybernetics*, *V*:254 – 259 (1993).

37. Fleurent, C. and Ferland, J. A. "Genetic hybrids for the Quadratic Assignment Problem," *DIMACS Series in Mathematics and Theoretical Computer Science*, *16*:190 – 206 (1994).

38. Fogel, D. B. and Angeline, P. J. "Guidelines for a suitable encoding." *Evolutionary Computation 1: Basic Algorithms and Operators* edited by T. Bäck, et al., chapter 8, 160–165, Bristol: Institute of Physics Publishing, 2000.

39. Fonseca, C. M. and Fleming, P. J. "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization." *Proceedings of the Fifth International Conference on Genetic Algorithms*, edited by Stephanie Forrest. 416–423. San Mateo, California: Morgan Kauffman Publishers, 1993.

40. Fonseca, C. M. and Fleming, P. J. *Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms I: A Unified Formulation*. Technical Report 564, Sheffield, UK: University of Sheffield, January 1995.

41. Fonseca, C. M. and Fleming, P. J. "An Overview of Evolutionary Algorithms in Multiobjective Optimization," *Evolutionary Computation*, *3*(1):1–16 (Spring 1995).

42. Fonseca, C. M. and Fleming, P. J. "Multiobjective Optimization." *Handbook of Evolutionary Computation 1*, edited by Thomas Bäck, et al., C4.5:1–C4.5:9, Institute of Physics Publishing and Oxford University Press, 1997.

43. Fujisawa, K. and Kubo, M. "Experimental Analyses of the Life Span Method for the Quadratic Assignment Problem," *The Institute of Statistical Mathematics Cooperative Research Report*, *75*:166 – 188 (1995).

44. Gambardella, L. M., Taillard, E. D., and Dorigo, M. "Ant Colonies for the Quadratic Assignment Problems," *Journal of the Operational Research Society*, *50*:167–176 (1999).

45. Goldberg, D. E. "Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis," *Complex Systems*, *3*:153 – 171 (1989).

46. Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1989.

47. Goldberg, D. E., Deb, K., and Clark, J. H. "Genetic Algorithms, Noise, and the Sizing of Populations," *Complex Systems*, *6*:333–362 (1992).

48. Goldberg, D. E., Deb, K., Kargupta, H., and Harik, G. "Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms." *Proceedings of the Fifth International Conference on Genetic Algorithms*, edited by Stephanie Forrest. 56–64. Morgan Kauffmann Publishers, 1993.

49. Goldberg, D. E., Deb, K., and Korb, B. "Messy Genetic Algorithms Revisited: Studies in Mixed Size and Scale," *Complex Systems*, *4*:415 – 444 (1990).

50. Goldberg, D. E., Korb, B., and Deb, K. "Messy Genetic Algorithms: Motivation, Analysis, and First Results," *Complex Systems*, *3*:493–530 (1989).

51. Grama, A., Gupta, A., Karypis, G., and Kumar, V. *Introduction to Parallel Computing - Second Edition*. Boston, MA: Addison-Wesly, 2003.

52. Hahn, P., Hall, N., and Grant, T. "A Branch-and-Bound Algorithm for the Quadratic Assignment Problem Based on the Hungarian Method," *European Journal of Operational Research* (August 1998).

53. Hart, E. and Ross, P. "GAVEL - a new tool for genetic algorithm visualization," *IEEE Transactions on Evolutionary Computation*, *5*(4):335–348 (August 2001).

54. Higuchi, T. and Manderick, B. "Hardware Realizations of Evolutionary Algorithms." *Evolutionary Computation 2: Advanced Algorithms and Operators* edited by T. Bäck, et al., chapter 26, 253–264, Bristol: Institute of Physics Publishing, 2000.

55. Hines, J., Thorpe, J. T., Kenneth B. Winiecki, J., and Frederick C. Harris, J. "Solving Quadratic Assignment Problems with Parallel Genetic Algorithms." *Proceedings of the ISCA International Conference*. 11 – 15. 1995.

56. Horn, J. and Nafpliotis, N. *Multiobjective Optimization using the Niched Pareto Genetic Algorithm*. Technical Report IlliGAl Report 93005, Urbana, Illinois, USA: University of Illinois at Urbana-Champaign, 1993.

57. Horn, J., Nafpliotis, N., and Goldberg, D. E. "A Niched Pareto Genetic Algorithm for Multiobjective Optimization." *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence1*. 82–87. Piscataway, New Jersey: IEEE Service Center, June 1994.

58. Horng, J.-T., Chen, C.-C., Liu, B.-J., and Kao, C.-Y. "Resolution of quadratic assignment problems using an evolutionary algorithm." *Proceedings of the 2000 Congress on Evolutionary Computation2*. 902–909. IEEE, 2000.

59. Hwang, K. and Xu, Z. *Scalable Parallel Computing: Technology, Architecture, Programming*. New York, NY: McGraw-Hill, 1998.

60. Ishibuchi, H. and Murata, T. "Multi-Objective Genetic Local Search Algorithm." *Proceedings of the 1996 International Conference on Evolutionary Computation*, edited by Toshio Fukuda and Takeshi Furuhashi. 119–124. Nagoya, Japan: IEEE, 1996.

61. Ishii, S. and Niitsuma, H. "λ-opt neural networks for quadratic assignment problem," *ICANN99: Artificial Neural Networks (ICANN99)*, *1*:115 – 120 (1999).

62. Kaashoek, J. F. and Paelinck, J. H. P. *A Bilinear Programming Solution to the Quadratic Assignment Problem*. Technical Report 179, Econometric Institute Report from Erasmus University, 1999.

63. Kargupta, H. *SEARCH, Polynomial Complexity, and the Fast Messy Genetic Algorithm*. PhD dissertation, University of Illinois, October 1995.

64. Khan, N. *Bayesian Optimization Algorithms for Multiobjective and Heirarchically Difficult Problems*. MS thesis, University of Illinois at Urbana-Champaign, Urbana, IL, July 2003.

65. Kita, H., Yabumoto, Y., Mori, N., and Nishikawa, Y. "Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm." *Parallel Problem Solving from Nature—PPSN IV*, Lecture Notes in Computer Science, edited by Hans-Michael Voigt, et al. 504–512. Berlin, Germany: Springer-Verlag, September 1996.

66. Kleeman, M. P., Day, R. O., and Lamont, G. B. "Multi-Objective Evolutionary Search Performance with Explicit Building-Block Sizes for NPC Problems." *To appear*

in Congress on Evolutionary Computation (CEC'2004)4. Piscataway, New Jersey: IEEE Service Center, May 2004.

67. Knarr, M. R., Goltz, M. N., Lamont, G. B., and Huang, J. "*In Situ* Bioremediation of Perchlorate-Contaminated Groundwater using a Multi-Objective Parallel Evolutionary Algorithm." *Congress on Evolutionary Computation (CEC'2003)1*. 1604–1611. Piscataway, New Jersey: IEEE Service Center, December 2003.

68. Knjazew, D. and Golberg, D. E. "OMEGA - Ordering Messy GA: Solving Permutation Problems with the Fast Messy Genetic Algorithm and Random Keys." *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, edited by Darrell Whitley, et al. 181–188. Las Vegas, Nevada, USA: Morgan Kaufmann, 10-12 July 2000.

69. Knowles, J. and Corne, D. "M-PAES: A Memetic Algorithm for Multiobjective Optimization." *2000 Congress on Evolutionary Computation1*. 325–332. Piscataway, New Jersey: IEEE Service Center, July 2000.

70. Knowles, J. and Corne, D. *Instance Generators and Test Suites for the Multiobjective Quadratic Assignment Problem*. Technical Report TR/IRIDIA/2002-25, IRIDIA, 2002. (Accepted for presentation/publication at the *2003 Evolutionary Multi-criterion Optimization Conference (EMO-2003)), Faro, Portugal.*

71. Knowles, J. and Corne, D. "Towards Landscape Analyses to Inform the Design of Hybrid Local Search for the Multiobjective Quadratic Assignment Problem." *Soft Computing Systems: Design, Management and Applications*, edited by A. Abraham, et al. 271–279. Amsterdam: IOS Press, 2002. ISBN 1-58603-297-6.

72. Knowles, J. and Corne, D. "Instance Generators and Test Suites for the Multiobjective Quadratic Assignment Problem." *Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2003, Faro, Portugal, April 2003, Proceedings*, number 2632 in LNCS, edited by Carlos Fonseca, et al. 295–310. Springer, 2003.

73. Knowles, J. D. and Corne, D. W. "The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation." *1999 Congress on Evolutionary Computation*. 98–105. Washington, D.C.: IEEE Service Center, July 1999.

74. Knowles, J. D. and Corne, D. W. "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy," *Evolutionary Computation*, *8*(2):149–172 (2000).

75. Knowles, J. D., Oates, M. J., and Corne, D. W. "Multiobjective Evolutionary Algorithms Applied to two Problems in Telecommunications," *BT Technology Journal*, *18*(4):51–64 (October 2000).

76. Koopmans, T. C. and Beckmann, M. "Assignment Problems and the Location of Economic Activities," *Econometrica*, *25*(1):53–76 (January 1957).

77. Kubo, M. and Fujisawa, K. "The Life Span Method – A New Variant of Local Search," *Japan Journal of Industrial and Applied Mathematics*, *15*(3):363–394 (October 1998).

78. Kumar, R., Parida, P. P., and Gupta, M. "Topological Design of Communication Networks using Multiobjective Genetic Optimization." *Congress on Evolutionary Computation (CEC'2002)1*. 425–430. Piscataway, New Jersey: IEEE Service Center, May 2002.

79. Kumar, V. *Introduction to Parallel Computing* (1st Edition). Redwood City: Benjamin/Cummings Publishing Company, Inc, 1994.

80. Lamont, G. B., "Genetic Algorithm Global Search." CSCE 686 Class lecture notes.

81. Li, Y., Pardalos, P. M., and Resende, M. G. C. "A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem," *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, *16*:237 –261 (1994). In Quadratic Assignment and Related Problems.

82. Lin, S.-S. and Lin, F.-C. "A General Interconnection Network with the Consideration of Locality in Traffic." *Proceedings of the 1989 International Symposium on VLSI Technology, Systems and Applications*. 297 – 301. 1989.

83. Lopez, R., "The Revolution will not be Piloted," June 2003. Popular Science, 2003, `http://www.popsci.com/popsci/aviation/article/0,12543,452052-1,00.html`; accessed February 27, 2004.

84. Maniezzo, V. and Colorni, A. "The ant system applied to the quadratic assignment problem," *IEEE Transactions on Knowledge and Data Engineering*, *11*:769–778 (1999).

85. Merkle, L. D. *Generalization and Parallelization of Messy Genetic Algorithms and Communication in Parallel Genetic Algorithms*. Thesis, Air Force Institute of Technology, 1992.

86. Merkle, L. D. *Analysis of Linkage-Friendly Genetic Algorithms*. Dissertation, Air Force Institute of Technology, Wright Patterson AFB, OH, 1996.

87. Merz, P. and Freisleben, B. "A Genetic Local Search Approach to the Quadratic Assignment Problem." *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, edited by Thomas Bäck. 465 – 472. San Francisco, CA: Morgan Kaufmann, 1997.

88. Merz, P. and Freisleben, B. "A Comparison of Memetic Algorithms, Tabu Search, and Ant Colonies for the Quadratic Assignment Problem." *Proceedings of the 1999 Congress on Evolutionary Computation3*, edited by Peter J. Angeline, et al. 2063 – 2070. Mayflower Hotel, Washington D.C., USA: IEEE Press, 1999.

89. Merz, P. and Freisleben, B. "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Transactions on Evolutionary Computation*, *4*:337–352 (2000).

90. Michaud, S. R. *Solving the Protein Structure Prediction Problem with Parallel Messy Genetic Algorithms*. MS thesis, Air Force Institute of Technology, Wright Patterson AFB, OH, March 2001.

91. Nissen, V. "Solving the quadratic assignment problem with clues from nature," *IEEE Transactions on Neural Networks*, *5*:66–72 (1994).

92. Nissen, V. "Quadratic assignment." *Handbook of Evolutionary Computation* edited by Thomas B"ack, et al., Institute of Physics Publishing and Oxford University Press, 1997.

93. Nyström, M. *Solving Certain Large Instances of the Quadratic Assignment Problem: Steinberg's Examples*. Technical Report CaltechCSTR:2001.010, Caltech, 1999.

94. Pardalos, P. M., Rendl, F., and Wolkowicz, H. "The Quadratic Assignment Problem: A Survey and Recent Developments." *Proceedings of the DIMACS Workshop on Quadratic Assignment Problems16*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 1 – 41. American Mathematical Society, 1994.

95. Pardalos, P. M. and Wolkowicz, H. "Quadratic Assignment and Related Problems." *Proceedings of the DIMACS Workshop on Quadratic Assignment Problems*, edited by Panos M. Pardalos and Henry Wolkowicz. 1994.

96. Pelikan, M. and Goldberg, D. E. "Heirarchical Problem Solving and the Bayesian Optimization Algorithm." *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. 267–274. 2000.

97. Pike, J., "Unmanned Aerial Vehicles (UAVs)." Federation of American Scientists, 2002, `http://www.fas.org/irp/program/collect/uav.htm`; accessed November 12, 2002.

98. Ramakrishnan, K. G., RESENDE, M. G. C., and PARDALOS, P. M. "A branch and bound algorithm for the quadratic assignment problem using a lower bound based on linear programming." *State of the Art in Global Optimization: Computational Methods and Applications*, edited by C. Floudas and P. M. PARDALOS. Kluwer Academic Publishers, 1995.

99. Sait, S. M., Youssef, H., and Khan, J. A. "Fuzzy Evolutionary Algorithm for VLSI Placement." *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, edited by Lee Spector, et al. 1056–1063. San Francisco, California: Morgan Kaufmann Publishers, 2001.

100. Sait, S. M., Youssef, H., and Khan, J. A. "Fuzzy Evolutionary Algorithm for VLSI Placement." *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, edited by Lee Spector, et al. 1056–1063. San Francisco, California, USA: Morgan Kaufmann, 7-11 July 2001.

101. Sastry, S. S., "DARPA SEC Kickoff," August 1998. University of California, Berkeley, 1998, `http://sec.eecs.berkeley.edu/talks/99/kickoff2/kickoff2.pdf`; accessed February 27, 2004.

102. Schaffer, J. D. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD dissertation, Vanderbilt University, 1984.

103. Schaffer, J. D. "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms." *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*. 93–100. Lawrence Erlbaum, 1985.

104. Schaffer, J. D. and Grefenstette, J. J. "Multiobjective Learning via Genetic Algorithms." *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*. 593–595. Los Angeles, California: AAAI, 1985.

105. Schott, J. R. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*. MS thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

106. Sim, K. M. and Sun, W. H. "Multiple Ant-Colony Optimization for Network Routing." *First International Symposium on Cyber Worlds (CW'02)2241*. 277–281. IEEE, 2002.

107. Smith, K. "Solving the Generalised Quadratic Assignment Problem using a Self-Organising Process." *Proceedings of the IEEE International Conference on Neural Networks (ICNN)4*. 1876 – 1879. IEEE Press, 1995.

108. Space Daily, , "Global Hawk UAV Completes 1,000 Hours Of Combat Support." Space Daily, 2002, `http://www.spacedaily.com/news/uav-02q.html`; accessed November 12, 2002.

109. Srinivas, N. and Deb, K. "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation*, *2*(3):221–248 (Fall 1994).

110. Srinivasan, A., Ramakrishnan, K. G., Kumaran, K., Aravamudan, M., and Naqvi, S. A. "Optimal Design of Signaling Networks for Internet Telephony," *IEEE Conference on Computer Communications (INFOCOM)*, (2000):707–716 (2000).

111. Taillard, Éric D.. and Gambardella, L. M. "An Ant Approach for Structured Quadratic Assignment Problems," *Second Metaheuristic International Conference*, 217 – 222 (July 1997).

112. Tang, K. S., Man, K. F., and Ko, K. T. "Wireless LAN Design using Hierarchical Genetic Algorithm." *Proceedings of the Seventh International Conference on Genetic Algorithms*, edited by Thomas Bäck. 629–635. San Mateo, California: Morgan Kaufmann Publishers, July 1997.

113. Tang, K.-S., Man, K.-F., and Kwong, S. "Wireless Communication Network Design in IC Factory," *IEEE Transactions on Industrial Electronics*, *48*(2):452–459 (April 2001).

114. Tate, D. M. and Smith, A. E. "A Genetic Approach to the Quadratic Assignment Problem," *Computers & Operations Research*, *22*(1):73 – 83 (1995).

115. The Economist, , "Look, no pilot." The Economist, 2000, `http://www.economist.com/science/displayStory.cfm?Story_ID=417775`; accessed October 10, 2002.

116. Thonemann, U. W. "Finding Improved Simulated Annealing Schedules with Genetic Programming." *Proceedings of the 1994 IEEE World Congress on Computational Intelligence1*. 391 – 395. Orlando, Florida, USA: IEEE Press, June 1994.

117. Van Veldhuizen, D. A. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD dissertation, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

118. Van Veldhuizen, D. A. and Lamont, G. B. "Evolutionary Computation and Convergence to a Pareto Front." *Late Breaking Papers at the Genetic Programming 1998 Conference*, edited by John R. Koza. 221–228. Stanford University, California: Stanford University Bookstore, July 1998.

119. Van Veldhuizen, D. A. and Lamont, G. B. "On Measuring Multiobjective Evolutionary Algorithm Performance." *2000 Congress on Evolutionary Computation1*. 204–211. Piscataway, New Jersey: IEEE Service Center, July 2000.

120. Van Veldhuizen, D. A., Sandlin, B. S., , Marmelstein, R. M., and Lamont, G. B. "Finding Improved Wire-Antenna Geometries with Genetic Algorithms." *Proceedings of the 1998 International Conference on Evolutionary Computation*, edited by David B. Fogel. 102–107. Piscataway, New Jersey: IEEE, 1998.

121. Vázquez, M. and Whitley, L. D. "A Hybrid Genetic Algorithm for the Quadratic Assignment Problem." *GECCO 2000*. 135 – 142. 2000.

122. Weiker, N., Szabo, G., Weiker, K., and Widmayer, P. "Evolutionary Multiobjective Optimization for Base Station Transmitter Placement With Frequency Assignment." *IEEE Tansactions on Evolutionary Computation7*. 189–203. IEEE, April 2003.

123. Weile, D. S. and Michielssen, E. *Integer coded Pareto genetic algorithm design of constrained antenna arrays*. Technical Report CCEM-13-96, Electrical and Computer Engineering Department, Center for Computational Electromagnetics, University of Illinois at Urbana-Champaign, November 1996.

124. Wess, B. and Gotschlich, M. "Optimal DSP Memory Layout Generation as a Quadratic Assignment Problem." *1997 IEEE International Symposium on Circuit and Systems*. 1712 – 1715. Hong Kong: IEEE PRess, June 1997.

125. Yamada, S. "A New Formulation of the Quadratic Assignment Problem on r-Dimensional Grid," *IEEE Transactions on Circuits and Systems*, *39*(10):791 –797 (October 1992).

126. Yip, P. P. C. and Pao, Y.-H. "A Guided Evolutionary Simulated Annealing Approach to the Quadratic Assignment Problem," *IEEE Transactions on Systems, Man, and Cybernetics*, *24*(9):1383 – 1387 (September 1994).

127. Zitzler, E. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

128. Zitzler, E., Deb, K., and Thiele, L. "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results," *Evolutionary Computation*, *8*(2):173–195 (Summer 2000).

129. Zitzler, E., Laumanns, M., and Thiele, L. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm." *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, edited by K. Giannakoglou, et al. September 2001.

130. Zitzler, E. and Thiele, L. *An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach*. Technical Report 43, Zurich, Switzerland: Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), May 1998.

131. Zitzler, E. and Thiele, L. "Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study." *Parallel Problem Solving from Nature V*, edited by A. E. Eiben. 292–301. Amsterdam: Springer-Verlag, September 1998.

132. Zitzler, E. and Thiele, L. "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, *3*(4):257–271 (November 1999).

133. Zydallis, J. *Explicit Building-Block Multiobjective Genetic Algorithms: Theory, Analysis, and Development*. PhD dissertation, Air Force Institute of Technology, Wright Patterson AFB, OH, March 2003.

134. Zydallis, J. B. and Lamont, G. B. "Explicit Building-Block Multiobjective Evolutionary Algorithms for NPC Problems." *Congress on Evolutionary Computation (CEC'2003)4*. 2685–2695. Piscataway, New Jersey: IEEE Service Center, December 2003.

135. Zydallis, J. B., Veldhuizen, D. A. V., and Lamont, G. B. "A Statistical Comparison of Multiobjective Evolutionary Algorithms Including the MOMGA–II." *First International Conference on Evolutionary Multi-Criterion Optimization* edited by Eckart Zitzler, et al., 226–240, Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
23-03-2004	**Master's Thesis**	Aug 2003 – Mar 2004

4. TITLE AND SUBTITLE

OPTIMIZATION OF HETEROGENEOUS UAV COMMUNICATIONS USING THE MULTIOBJECTIVE QUADRATIC ASSIGNMENT PROBLEM

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Kleeman, Mark, P., 1st Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCE/ENG/04-04

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/IFTA (AFMC)
Attn: Dr. Robert L. Ewing
2241 Avionics, Bldg 620, Rm S3-A52 DSN: 785-6635, ext. 3592
WPAFB, OH 45433 e-mail: Robert.Ewing@wpafb.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Air Force has placed a high priority on developing new and innovative ways to use Unmanned Aerial Vehicles (UAVs). The Defense Advanced Research Projects Agency (DARPA) currently funds many projects that deal with the advancement of UAV research. The ultimate goal of the Air Force is to use UAVs in operations that are highly dangerous to pilots, mainly the suppression of enemy air defenses (SEAD). With this goal in mind, formation structuring of autonomous or semi-autonomous UAVs is of future importance.

This particular research investigates the optimization of heterogeneous UAV multi-channel communications in formation. The problem maps to the multiobjective Quadratic Assignment Problem (mQAP). Optimization of this problem is done through the use of a Multiobjective Evolutionary Algorithm (MOEA) called the Multiobjective Messy Genetic Algorithm - II (MOMGA-II). Experimentation validates the attainment of an acceptable Pareto Front for a variety of mQAP benchmarks. It was observed that building block size can affect the location vectors along the current Pareto Front. The competitive templates used during testing perform best when they are randomized before each building block size evaluation. This tuning of the MOMGA-II parameters creates a more effective algorithm for the variety of mQAP benchmarks, when compared to the initial experiments. Thus this algorithmic approach would be useful for Air Force decision makers in determining the placement of UAVs in formations.

15. SUBJECT TERMS

Unmanned, Aerial Reconnaissance, Aerial Warfare, Formation Flight, and Wireless Communications

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Gary B. Lamont, AD-24, DAF
U	U	U	UU	192	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4718; e-mail: gary.lamont@afit.edu